Red Hat Reference Architecture Series

Deploying Highly Available NFS on Red Hat Enterprise Linux 6

Scott Collier, RHCA

Principal Software Engineer

Version 1.0

July 2011

redhat.

1801 Varsity Drive™
Raleigh NC 27606-2072 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588
Research Triangle Park NC 27709 USA

The information contained herein is subject to change without notice. Red Hat, Inc. shall not be liable for technical or editorial errors or omissions contained herein.

The GPG fingerprint of the security@redhat.com key is:
CA 20 86 86 2B D6 9D FC 65 F6 EC C4 21 91 80 CD DB 42 A6 0E

Send feedback to refarch-feedback@redhat.com

Table of Contents

1 Executive Summary

As data availability becomes more and more important, reducing the single points of failure in any environment becomes a key factor to reduce down time. This reference architecture demonstrates how to obtain Red Hat's High Availability Add-On (software) for Red Hat Enterprise Linux 6 as well as how the installation and configuration process works. File system availability is a core component of any information technology infrastructure. The example used in this reference architecture is how to set up and configure a Network File System (NFS) service that can automatically relocated from one node to the other to ensure high availability.

This paper contains two demonstrations on how to configure this solution. The first method depicts the use of *Conga* – the graphical cluster configuration utility. The second method uses Cluster Configuration System (CCS) – the command line configuration utility. Depending on the environment either style of management may be chosen.

This reference architecture provides three administration examples that can be performed on a Red Hat Enterprise Linux highly available cluster. First will be importing a CCS created cluster into the *luci* management interface. Second will be adding additional NFS clients to the cluster services. Finally, best practices will be provided on how to properly update a Red Hat Enterprise Linux cluster.

2 Solution Overview

This section provides an overview of the Red Hat Enterprise Linux operating system, Red Hat's High Availability Add-On components, and the file systems involved in this reference architecture.

2.1 Red Hat Enterprise Linux 6

Red Hat Enterprise Linux 6.1, the latest release of Red Hat's trusted datacenter platform, delivers advances in application performance, scalability, and security. With Red Hat Enterprise Linux 6.1; physical, virtual, and cloud computing resources can be deployed within the data center.

Reliability, availability, and security (RAS):

- CPU and memory hot-add for Nehalem-EX based systems, with proper hardware support
- RAS hardware-based hot adding of CPUs and memory is enabled
- Memory pages with errors can be declared as “poisoned” and will be avoided

File Systems:

- *ext4* is the default file system and scales to 16TB
- *XFS* is available as an Add-On and can scale to 100TB
- *Fuse* allows file systems to run in user space which allows testing and development on newer fuse-based file systems (such as cloud file systems)

High Availability:

- Extends the current clustering solution to the virtual environment allowing for high availability of virtual machines and applications running inside those virtual machines
- Enables Network File System version 4 (NFSv4) resource agent monitoring
- Introduction of CCS. CCS is a command line tool that allows for complete CLI administration of the Red Hat's High Availability Add-On

Resource Management:

- *cgroups* organize system tasks so that they can be tracked thus allowing other system services the ability to control the resources that *cgroup* tasks may consume
- *cpuset* applies CPU resource limits to *cgroups,* allowing processing performance to be allocated to tasks

There are many other feature enhancements to Red Hat Enterprise Linux 6. Please see the Red Hat[1] website for more information.

2.2 Red Hat Network Satellite Server

This reference architecture uses Red Hat Network Satellite (RHN) server to provision the cluster nodes. All RHN functionality is on the network, allowing much greater functionality and customization. The satellite server connects with Red Hat over the public Internet to download new content and updates. This model also allows customers to take their Red Hat Network solution completely off-line if desired. Features include:

- Support for provisioning Red Hat Enterprise Linux 6
- Compliance features SELinux context deployment and reporting
- API layer allows the creation of scripts to automate functions or integrate with existing management applications.
- External repository synchronization
- Create staged environments (development, test, production) to select, manage and test content in a structured manner.
- The ability to remove duplicate profiles
- Access to advanced features in the Provisioning Module, such as bare metal PXE boot provisioning and integrated network install trees.
- Access to Red Hat Network Monitoring Module for tracking system and application performance.

RHN Satellite is Red Hat’s on-premise systems management solution that provides software updates, configuration management, provisioning and monitoring across both physical and virtual Red Hat Enterprise Linux servers. It offers customers opportunities to gain enhanced performance, centralized control and higher scalability for their systems, while deployed on a management server located inside the customer’s data center and firewall.

Red Hat released RHN Satellite 5.4.1 in June 2011. This version offers opportunities for increased flexibility and faster provisioning setups for customers with the incorporation of open source Cobbler technology in its provisioning architecture.

2.3 High Availability Add-On Components

2.3.1 Quorum

The *quorum* is a voting algorithm used by the cluster manager (CMAN). CMAN manages cluster quorum and cluster membership. CMAN runs as a service on all the cluster nodes. To maintain *quorum* the nodes in the cluster must agree about their status among themselves. The *quorum* determines which nodes in the cluster are dominant. For example, if there are three nodes in a cluster and one node loses connectivity, the other two nodes communicate with each other and determine that the third node needs to be fenced. The action of fencing ensures that the node which lost connectivity does not corrupt data.

By default each node in the cluster has one *quorum* vote, although this is configurable. There are two methods the nodes can communicate with each other to determine *quorum.* The first method is via Ethernet. For *quorum* via network, quorum consists of a simple majority (50% of the nodes +1 extra). The second method is by adding a *quorum* disk. The *quorum* disk allows for user-specified conditions to exist which help determine the majority.

2.3.2 Resource Group Manager

The resource group manager (*rgmanager*) provides failover capabilities for collections of cluster resources, resource groups, or resource trees. *rgmanager* works by allowing systems administrators to define, configure, and monitor cluster services. In the event of a node failure *rgmanager* relocates the clustered service to another node with minimal service disruption. Services can be restricted to certain nodes, such as restricting *httpd* to one group of nodes while *mysql* can be restricted to a separate set of nodes.

There are various processes and agents that constitute *rgmanager* . The following list summarizes those areas.

- Failover Domains - An ordered subset of members to which a service may be bound
- Service Policies - *rgmanager's* service startup and recovery policies
- Resource Trees - Representations of resources, their attributes, parent / child and sibling relationships
- Service Operational Behaviors - Five available actions that a user may invoke to manage the services: enable, disable, relocate, stop and migrate
- Virtual Machine Behaviors – Behaviors to remember when running virtual machines in a *rgmanager* cluster
- Resource Actions - Actions that rgmanager uses and how to customize their behavior from the */etc/cluster/cluster.conf* file
- Event Scripting - If *rgmanager's* failover and recovery policies do not fit the environment, customize using the scripting subsystem

rgmanager runs as a service on all the nodes in a cluster. If the service is not running, the resources are not available to be brought online. Recovery of *rgmanager* depends on the Distributed Lock Manager (DLM). In the event of a failure, the DLM must recover prior to *rgmanager* recovering services from a failed host.

2.3.3 Fencing

Fencing is the disconnection of a node from the cluster's shared storage. Fencing prevents the affected node from issuing I/O to shared storage, thus ensuring data integrity. The cluster infrastructure performs fencing through *fenced,* the fence daemon**.**

When CMAN determines that a node has failed, it communicates to other cluster-infrastructure components to inform them that the node has failed. The failed node is fenced when *fenced* is notified. Other cluster-infrastructure components determine what actions to take — that is, they perform any recovery that needs to done. For example, distributed lock manager (*DLM*) and Global File System version 2 (*GFS2)*, when notified of a node failure, suspend activity until they detect that **`fenced`** has completed fencing the failed node. Upon confirmation that the failed node is fenced, *DLM* and *GFS2* perform recovery. *DLM* releases locks of the failed node; *GFS2* recovers the journal of the failed node.

The fencing program determines from the cluster configuration file which fencing method to use. Two key elements in the cluster configuration file define a fencing method: fencing agent and fencing device. The fencing program makes a call to a fencing agent specified in the cluster configuration file. The fencing agent, in turn, fences the node via a fencing device. When fencing is complete, the fencing program notifies the cluster manager. The **High Availability Add-On** provides a variety of fencing methods:

- Power fencing — A fencing method that uses a power controller to power off an inoperable node.
- Storage fencing — A fencing method that disables the Fibre Channel port that connects storage to an inoperable node. SCSI 3 persistent reservations fencing can be used here as well.
- Systems management fencing — Several other fencing methods that disable I/O or power of an inoperable node, including IBM Bladecenters, PAP, DRAC/MC, HP ILO, IPMI, IBM RSA II, and others[2].

2.3.4 Failover Domains

A *failover domain* is an ordered subset of members to which a service may be bound. Failover domains, while useful for cluster customization, are not required for operation. The following is a list of semantics governing the options as to how the different configuration options affect the behavior of a failover domain.

Ordering, *restriction*, and *nofailback* are flags and may be combined in almost any way (e.g., *ordered+restricted*, *unordered+unrestricted*, etc.). These combinations affect both where services start after initial quorum formation and which cluster members take over services in the event that the service has failed.

- Preferred node or preferred member - The *preferred node* is the member designated to run a given service if the member is online. This behavior can be emulated by specifying an unordered, unrestricted failover domain of exactly one member.
- Restricted domain - Services bound to the domain may only run on cluster members

which are also members of the *failover* domain. If no members of the *failover* domain are available, the service is placed in the stopped state. In a cluster with several members, using a restricted *failover* domain can ease configuration of a cluster service (such as **httpd**), which requires identical configuration on all members that run the service. Instead of setting up the entire cluster to run the cluster service, set up only the members in the restricted *failover* domain that are associated with the cluster service.

- Unrestricted domain - The default behavior. Services bound to this domain may run on all cluster members, but runs on a member of the domain whenever one is available. This means that if a service is running outside of the domain and a member of the domain comes online, the service will migrate to that member.
- Ordered domain - The order specified in the configuration dictates the order of preference of members within the domain. The highest-ranking online member of the domain hosts the service . This means that if member A has a higher-rank than member B, the service relocates to A (if currently running on B) when A transitions from offline to online.
- Unordered domain - The default behavior. Members of the domain have no order of preference; any member may run the service. In an unordered domain, services always migrate to members of their *failover* domain whenever possible.
- Failback - Services on members of an ordered *failover* domain should relocate back to the original node once the failure is addressed. *Failback* can also be disabled, this is useful for frequently failing nodes to prevent frequent service shifts between the failing node and the *failover* node.

2.3.5 ACPI

Advanced Configuration and Power Interface (ACPI) disables the immediate shut off of a node. For example, when the power button is pushed on servers with certain hardware (depends on BIOS support), a system shut down t could take 4 – 5 seconds since ACPI attempts to shut down cleanly. In this reference architecture ACPI is disabled in the clustered environment to minimize the time of a shut down in order to prevent data corruption. There are three ways to disable ACPI:

- Disable ACPI Soft-Off if it is available in the server BIOS
- Append acpi=off to the kernel boot command in */boot/grub/grub.conf*
- Use **`chkconfig`** to disable *acpid*

The third method is the preferred way to disable ACPI. Once ACPI is disabled it can be tested by issuing the appropriate fence commands or by pushing the power button on the server.

2.3.6 Conga

A high level overview of some of the conga sub-components is shown in **Table 2.3.6-1: Cluster Components.** These components are only briefly described here as they are background processes that are automatically configured.

Service	Purpose
luci	Graphical User Interface
ricci	Cluster management and configuration agent

Table 2.3.6-1: Cluster Components

2.3.6.1 luci

Luci provides a web-based graphical user interface that helps visually organize the nodes in the cluster, manage fence devices, *failover* domains, resources, service groups, and other attributes of the cluster.

Luci can create a cluster from scratch or import an existing cluster. When an existing cluster is imported, all that is required is network access to a single node that is member of the cluster. *Luci* will then parse the */etc/cluster/cluster.conf* file and locate the other nodes in the cluster and then import them. *Luci* can manage multiple clusters as well.

Some of the improvements in the new version of *luci* are the redesigned interface and logging capabilities. For example, there are the global settings for logging and there are daemon specific log settings that cover *rgmanager*, *qdiskd*, *fenced*, *corosync*, *groupd* and more. *Luci* requires port 8084 open for proper connectivity.

2.3.6.2 Ricci

Ricci is the cluster management and configuration daemon that runs on the cluster nodes. When *ricci* is installed it creates a user account *ricci*. There must be a password set for the *ricci* user. This allows authentication from the *luci* server. Refer to the manual pages for more information on the *ricci* daemon.

Ricci dispatches incoming messages to the underlying management modules. Invoking *ricci* with no optionscauses ricci to operate as a daemon which listens on port 11111 by default. It is important to open up port 11111 for both tcp and udp in order for *ricci* to function properly[3].

2.3.7 CCS

Cluster Configuration System (CCS) was introduced in Red Hat Enterprise Linux 6.1. CCS provides a powerful new way of managing a Red Hat Enterprise Linux cluster. CCS allows an administrator to create, modify and view a cluster configuration file on a remote node through *ricci* or on a local file system. CCS has a robust man page utilizing the following switches for the `ccs` command as shown in **Table 2.3.7-1: Example CCS Switches**.

Switches	Function
-f	Configure local file
-h	Configure remote host
--checkconf	Verifies all nodes have same cluster.conf
--createcluster	Creates a cluster from scratch
--startall, --stopall	Starts and stops cluster services on all nodes
--addfenceinst	Adds a fence instance
--addservice	Adds a service to the cluster
--setlogging	Sets the logging options for the cluster
--lsnodes	Lists the nodes in the cluster

Table 2.3.7-1: Example CCS Switches

Once the cluster is created with CCS, the xml code in the *cluster.conf* file can be validated against the *cluster.rng* file provided with the CCS package using either of the following commands.

```
# xmllint \-\-relaxng /usr/share/ccs/cluster.rng cluster.conf
# ccs_config_validate
```

This ensures there are no problems with the xml before pushing it out to the cluster nodes.

CCS authenticates with *ricci* by using the automatically generated certificate files in *~/.ccs/.* These files allow CCS to communicate with *ricci* securely. To communicate with *ricci* the password for the *ricci* agent on each cluster node must be known by the administrator as CCS prompts for the password at the first connection.

This reference architecture demonstrates creating a cluster from scratch, adding the appropriate fencing mechanisms and services. Cluster management with **clusvcadm**, **clustat** and **ccs** is demonstrated to show how a cluster can be managed without a graphical user interface. CCS can be run on any node that has the **rhel-x86_64-server-ha-6** subscription with access to the cluster nodes.

2.4 File Systems

This Reference Architecture uses two file systems. The first is the file system that is used to format the block storage, *ext4.* The second is the Network File System (NFS v3) that shares out the local *ext4* file system to NFS clients. The *ext4* file system is new as of Red Hat 5.6 and is now the default file system for Red Hat Enterprise Linux.

2.4.1 NFS

The most common client/server file system for Red Hat Enterprise Linux customers is the Network File System (NFS). NFS is a very mature file system that is used in many Linux environments. Red Hat Enterprise Linux provides both an NFS server component that is used to export a local file system over the network and an NFS client that can be used to access these file systems.

Network file systems, also referred to as client/server file systems, allow client machines to access files that are stored on a server. This makes it possible for multiple users on multiple machines to share files and storage resources. Such file systems are built from one or more servers that export to one or more clients a set of file systems. The client nodes do not have direct access to the underlying block storage, but rather interact with the storage using a protocol that allows for better access control. Historically these systems have used layer 2 networking technologies like Gigabit Ethernet to provide reasonably good performance for a set of clients.

As NFS environments continue to grow and expand, many systems administrators want to avoid having a single point of failure in the environment. By introducing Red Hat High Availability into the solution with its native support for NFS this can be avoided. This paper demonstrates configuring an NFS share via the *conga* and CCS tools as well as how to relocate services and manage the NFS environment.

2.4.2 ext4

Ext4 is the successor to the E*xt3* file system. *Ext4* is POSIX compliant and provides support for several system calls such as *read()*, *write()* and *seek()*. *Ext4* is the fourth generation of the extended file system family and is the default file system in Red Hat Enterprise Linux 6. *Ext4* has been offered in test previews since Red Hat Enterprise Linux 5.3, giving customers confidence in its maturity. *Ext4* can read and write to *Ext2* or *Ext3* file systems, but Ext2 or *Ext3* can not read and write to an *Ext4* file system. However, *Ext4* adds several new and improved features that are common with most modern file systems, such as the following:

- Extent-based metadata
- Delayed allocation
- Journal check-summing
- Large storage support

A more compact and efficient way to track utilized space in a file system is the usage of extend-based metadata and the delayed allocation feature. These features improve file system performance and reduce the space consumed by metadata. Delayed allocation allows the file system to postpone selection of the permanent location for newly written user data until the data is flushed to disk. This enables higher performance since it can allow for larger, more contiguous allocations, allowing the file system to make decisions with much better information.

Additionally, file system repair time (fsck) in *Ext4* is much faster than in *Ext2* and *Ext3*. Some file system repairs have demonstrated up to a six-fold increase in performance. Currently, Red Hat’s maximum supported size for *Ext4* is 16TB in both Red Hat Enterprise Linux 5 and Red Hat Enterprise Linux 6. Application performance depends on many variables; in addition to the actual file system chosen, the application performance also depends on the specific I/O pattern the application generates and the type of server and storage hardware used.

2.4.3 XFS

XFS is a highly scalable, high-performance file system which was originally designed at Silicon Graphics, Inc. It was created to support extremely large filesystems (up to 16 exabytes), files (8 exabytes) and directory structures (tens of millions of entries).

XFS supports *metadata journaling*, which facilitates quicker crash recovery. The XFS file system can also be defragmented and enlarged while mounted and active. In addition, Red Hat Enterprise Linux 6 supports backup and restore utilities specific to XFS.

One example of how XFS can be tuned is that XFS can be optimized by ensuring the stripe geometry matches the underlying disk subsystem. This happens one of two ways. The first way is by providing mkfs.xfs the stripe unit and stripe width of the underlying storage array. For example, using a RAID array of 5 disks and a 512k chunk size, set the su to 5 (sw=5) and set the sw to 512k (su=512k). The format command would follow this format:

```
# mkfs.xfs -d su=512k,sw=5 /dev/mapper/EMCiSCSIp1
```

XFS supports other features as well, including

- Suspension of a XFS File System
- Backup and Restore
- Repairing
- Quotas
- More options

See the official XFS documentation for more information regarding the features and functionality of XFS on Red Hat Enterprise Linux.

3 Reference Architecture Configuration

This section provides an overview of the hardware that was used in this reference architecture. The image below shows the hardware setup using Dell PowerEdge servers and EMC storage as well as the networking configuration. This reference architecture contains two Dell PowerEdge R810's that are attached to a EMC Celerra NS-120FC array via iSCSI. The iSCSI LUN has a *ext4* file system created on it which is served out via the cluster using NFSv3. This configuration is Active-Passive, meaning that only one cluster node serves the file system out at a time. The management station running *luci* is a separate virtual machine. See **Illustration 3-1: Cluster Architecture**. The illustration shows the HA-NFS-Service which is composed of four resources: File System, NFS Export, NFS Client and finally the IP address shown at the bottom of the illustration.

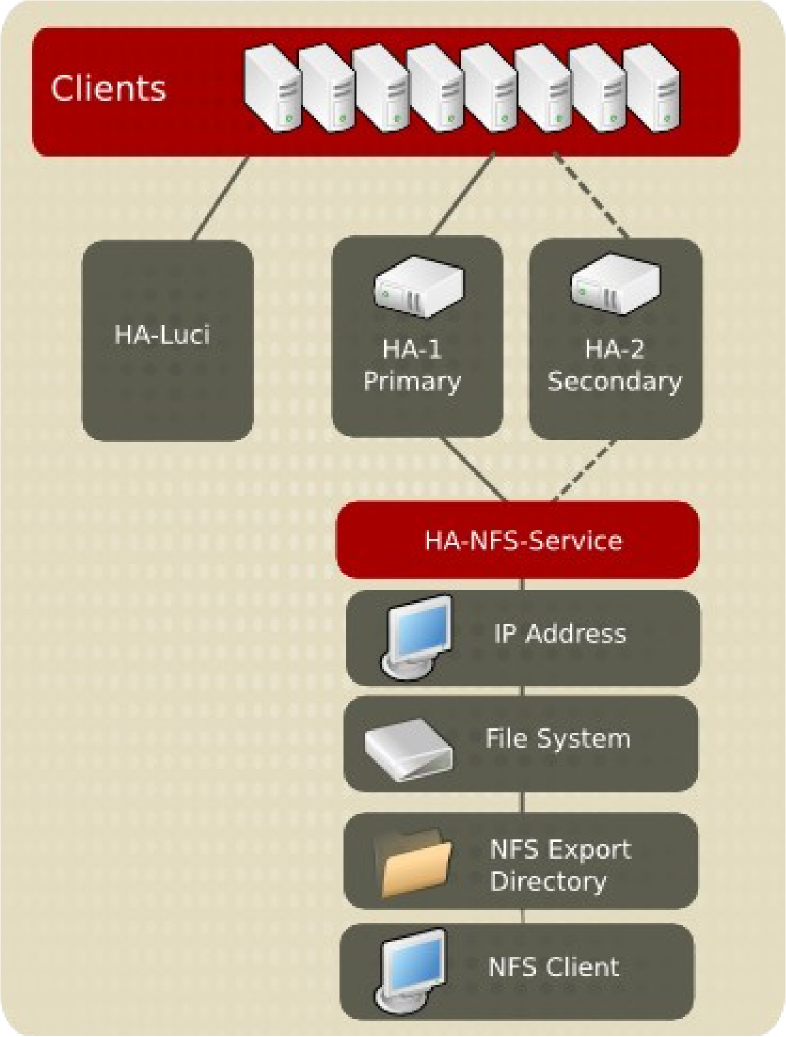


Illustration 3-1: Cluster Architecture

3.1 Hardware Configuration

This reference architecture has two physical servers (HA1, HA2) and one virtual server (HA-LUCI). See **Table 3.1-1: Server Hardware** for a listing of the systems. In addition to the server hardware, the storage and networking hardware are covered in **Table 3.1-2: Storage** and **Table 3.1-3: Ethernet Switches**.

Hardware Systems	Specifications
HA-LUCI [1 x KVM Virtual Server]	Dual Processors
	4GB Memory
	15GB HDD
HA{1,2} [2 x Dell PowerEdge R810s]	Dual Socket, 8 Core Processors Intel® Xeon® CPU X7560 @2.27GHz, 128GB RAM
	4 x 146 GB SAS internal disk drive (RAID 5)
	4x Broadcom NetXtreme II BCM 5709 1GB Ethernet Controllers
	2 x Broadcom NetXtreme II BCM57711 10Gb Ethernet Controller

Table 3.1-1: Server Hardware

The storage used in this reference architecture is a EMC Celerra NS-120FC as shown in **Table 3.1-2: Storage**. The Celerra is sharing out the LUN to the cluster nodes via iSCSI and securing the traffic by enabling Challenge Handshake Authentication Protocol (CHAP).

Hardware	Specifications
1 x EMC Celerra NS-120FC Fibre Channel and iSCSI Storage Array + CLARiiON CX4120 with Dual Storage Processors	NAS Version: 6.0.36-4
	CLARiiON Agent Revision: 7.30.0 (4.93)
	CLARiiON Code Revision: 04.30.000.5.004
	Expander Controller: Code Version: 1036
	iSCSI IP Addresses: 192.168.0.20, 192.168.0.21

Table 3.1-2: Storage

Hardware	Specifications
1 x HP ProCurve 5412zl Switch	Firmware: K.14.65, ROM K.12.21

Table 3.1-3: Ethernet Switches

3.2 Software Configuration

This reference architecture consists of the Red Hat Enterprise Linux software shown in **Table 3.2-1: Operating Systems** and the cluster software shown in **Table 3.2-2: Cluster Software** and **Table 3.2-3: Cluster Network**. This reference architecture also shows how to open ports using *iptables*. Refer to **Table 3.2-4: Cluster Management Node Ports4** and **Table 3.2-5: Cluster Node Ports** for a list of ports that must be open on each node.

Software	Role	Version
Red Hat Enterprise Linux (RHEL)	Cluster Nodes	6.1
Red Hat Enterprise Linux (RHEL)	Cluster Manager	6.1
Red Hat Enterprise Linux (RHEL)	KVM Host	6.0

Table 3.2-1: Operating Systems

Software	Version
CMAN (Cluster Manager)	3.0.12-41
iSCSI Initiator	6.2.0.872-21
rgmanager	3.0.12-11
CCS	0.16.2-35

Table 3.2-2: Cluster Software

Server	Networks	Settings
HA1	Public / Cluster Traffic	10.16.139.40
HA1	ISCSI	192.168.0.29
HA2	Public / Cluster Traffic	10.16.139.41
HA2	iSCSI	192.168.0.28
Shared NFS Service IP	Public	10.16.139.46

Table 3.2-3: Cluster Network

Cluster Management Node Ports	Protocol	Function
8084	TCP	luci

Table 3.2-4: Cluster Management Node Ports[4]

Cluster Node Ports	Protocol	Function
88{8,9}	TCP / UDP	NFS Ports
89{0,1}	TCP / UDP	NFS Ports
111	TCP / UDP	NFS Ports
2049	TCP / UDP	Client NFS traffic
16851	TCP	Modclusterd
11111	TCP / UDP	ricci
540{4,5}	UDP	Corosync / cman
21064	TCP	Distributed Lock Manager

Table 3.2-5: Cluster Node Ports

4 Solution Deployment

This section covers the deployment tasks required to install, configure, and manage a highly available NFS service using the Red Hat High Availability Add-On (Software) for Red Hat Enterprise Linux. There are two methods available to configure the NFS service, the first is via Conga and the second is via CCS. The installation of the software is the same for both configuration methods.

The first part of this section covers the installation of the software on both the cluster nodes and the management node and the second part of this section covers configuring the high availability software with Conga and CCS. A high level overview of the steps required to complete the deployment are:

1. Deploy the cluster management node
2. Install the cluster management software
3. Secure the cluster management node
4. Deploy cluster nodes
5. Configure the storage on the cluster nodes
6. Install the cluster software on the nodes (Optional if using luci)
7. Secure the cluster nodes
8. Configure the cluster services
9. Test the configuration

4.1 General Tasks

Location: HA-LUCI

1. Install the management node with Red Hat Enterprise Linux 6.1

For this reference architecture, the management node which is running luci and CCS is installed on a virtual machine in a controlled lab environment. Install Red Hat Enterprise Linux according to the installation guide.[5]

2. Secure the Management Server. The management server needs to allow traffic to the luci server over port 8084.

3. Backup the current iptables configuration

```
# cp /etc/sysconfig/iptables{,.orig}
```

4. Here are the current **iptables** configuration for the lab system. While these are the defaults, it really does not matter. The important chains are the ones that follow and they should not effect the default rule set.

```
# iptables –numeric --verbose --list --line-numbers
```

```
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out source destination
1      155 11804 ACCEPT     all  --  *      *  0.0.0.0/0 0.0.0.0/0
state RELATED,ESTABLISHED
2        0     0 ACCEPT     icmp --  *      *  0.0.0.0/0 0.0.0.0/0
3        0     0 ACCEPT     all  --  lo     *  0.0.0.0/0 0.0.0.0/0
4        0     0 ACCEPT     tcp  --  *      *  0.0.00.0/  0.0.0.0/0
state NEW tcp dpt:22
5        8  3771 REJECT     all  --  *      *  0.0.0.0/0 0.0.0.0/0
reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source destination
1        0     0 REJECT     all  --  *      *  0.0.0.0/0   0.0.0.0/0
reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT 82 packets, 9576 bytes)
num   pkts bytes target     prot opt in     out     source destination
```

5. Create and forward traffic to a new chain

```
# iptables --new-chain ha-cluster
# iptables --insert INPUT --jump ha-cluster
```

6. Add the cluster specific rules to the ha-cluster chain

```
# iptables --append ha-cluster --proto tcp --dport 8084 --jump ACCEPT
```

7. Save the rules and ensure iptables starts automatically on boot

```
# service iptables save
# chkconfig iptables on
```

Assign the Red Hat Network channels to the cluster management node

1. When the system boots up run **`rhn_register`**
2. After the system is registered, update the system

```
# yum update
```

3. Log into RHN and assign the following channels to the cluster management server
   - RHEL Server High Availability (v. 6 for 64-bit x86_64)
   - RHN Tools for RHEL (v. 6 for 64-bit x86_64)
4. Confirm the channels are available

```
# yum repolist
```

Install and start the Cluster Management Software

1. Install the *High Availability Management* group packages via `yum`

```
# yum groupinstall "High Availability Management"
```

2. Start the *luci* service and ensure it starts on reboot

```
# service luci restart
Stop luci...                                               [  OK  ]
```

```
Start luci...                                               [  OK  ]
Point your web browser to https://ha-luci.cloud.lab.eng.bos.redhat.com:8084
(or equivalent) to access luci

# chkconfig luci on
```

3. Log into the *luci* interface using a web browser with root credentials and verify functionality

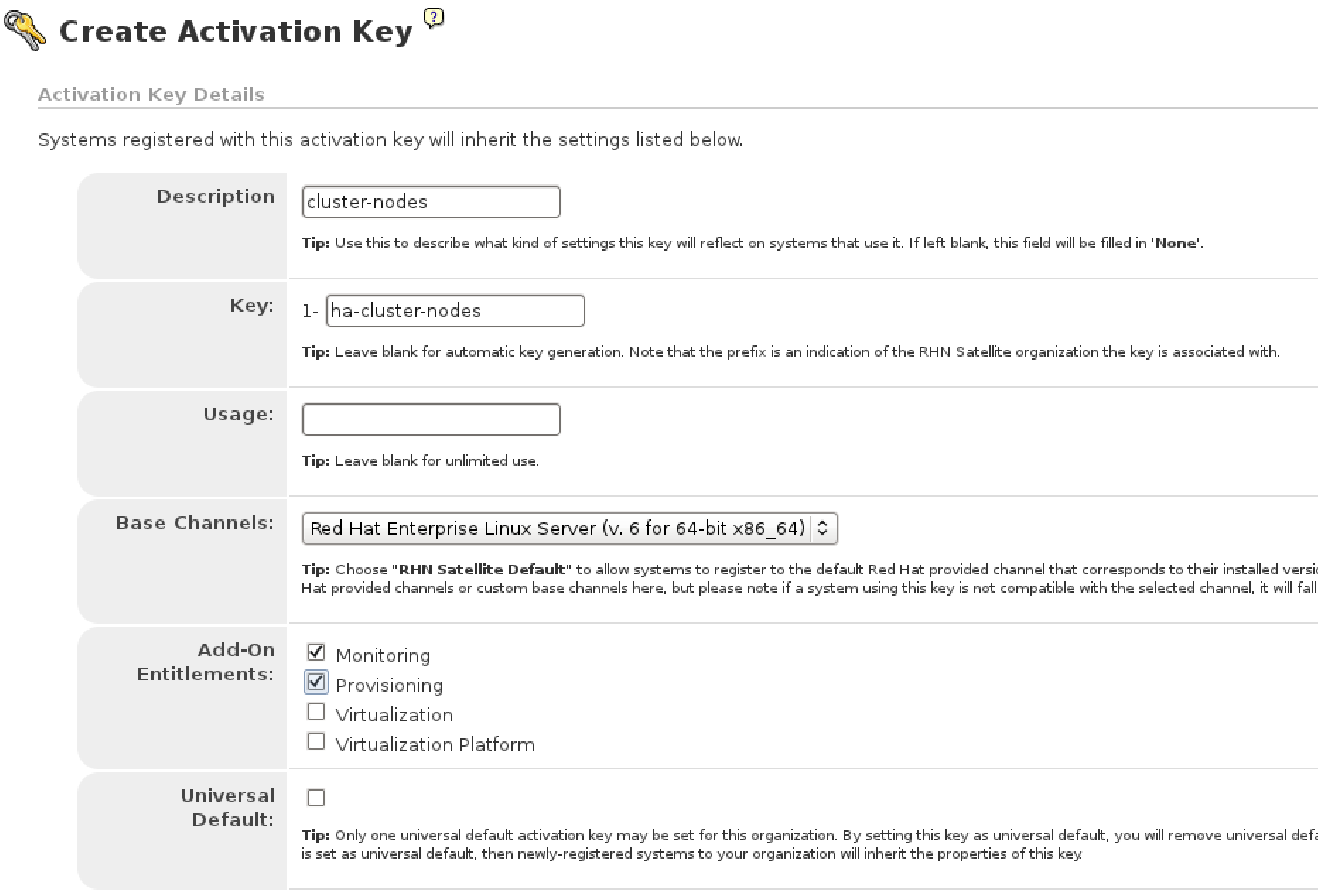
Another option is to automate this process by using Red Hat Network (RHN) Satellite server. For more details on installing and configuring Red Hat Enterprise Linux, please refer to the installation guides on the Red Hat[6] customer portal.

Install Red Hat Enterprise Linux 6.1 on the cluster nodes

To install Red Hat Enterprise Linux 6.1 on the cluster nodes RHN is used. An overview of the steps follows. Provisioning Tool: RHN Satellite

1. Create an Activation Key

   a) Click **Systems | Activation Key | Create New Key** and fill out Step 1. as shown in **Illustration 4.1-1: Create Activation Key**



Create Activation Key

Activation Key Details

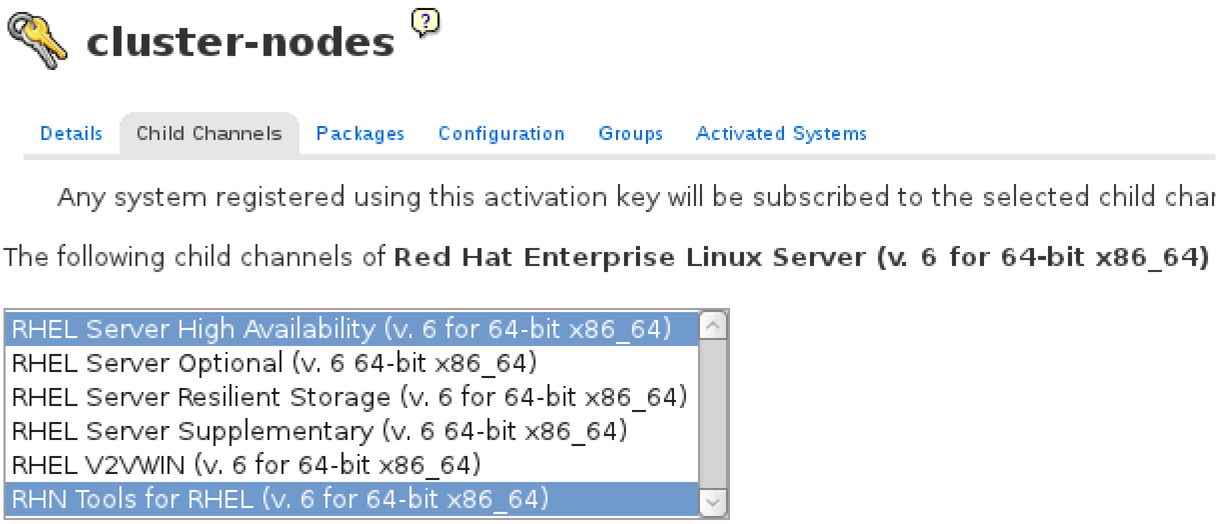
Systems registered with this activation key will inherit the settings listed below.

Description	cluster-nodes **Tip:** Use this to describe what kind of settings this key will reflect on systems that use it. If left blank, this field will be filled in '**None**'.
Key:	1- ha-cluster-nodes **Tip:** Leave blank for automatic key generation. Note that the prefix is an indication of the RHN Satellite organization the key is associated with.
Usage:	**Tip:** Leave blank for unlimited use.
Base Channels:	Red Hat Enterprise Linux Server (v. 6 for 64-bit x86_64) **Tip:** Choose "**RHN Satellite Default**" to allow systems to register to the default Red Hat provided channel that corresponds to their installed versi Hat provided channels or custom base channels here, but please note if a system using this key is not compatible with the selected channel, it will fall
Add-On Entitlements:	☑ Monitoring ☑ Provisioning ☐ Virtualization ☐ Virtualization Platform
Universal Default:	☐ **Tip:** Only one universal default activation key may be set for this organization. By setting this key as universal default, you will remove universal defa is set as universal default, then newly-registered systems to your organization will inherit the properties of this key.

Illustration 4.1-1: Create Activation Key

   b) Select ***Create Activation** Key*

2. Select the ***Child Channels*** tab and select the following channels as shown in **Illustration 4.1-2: Child Channels** and click **Update Key**



cluster-nodes

Any system registered using this activation key will be subscribed to the selected child char

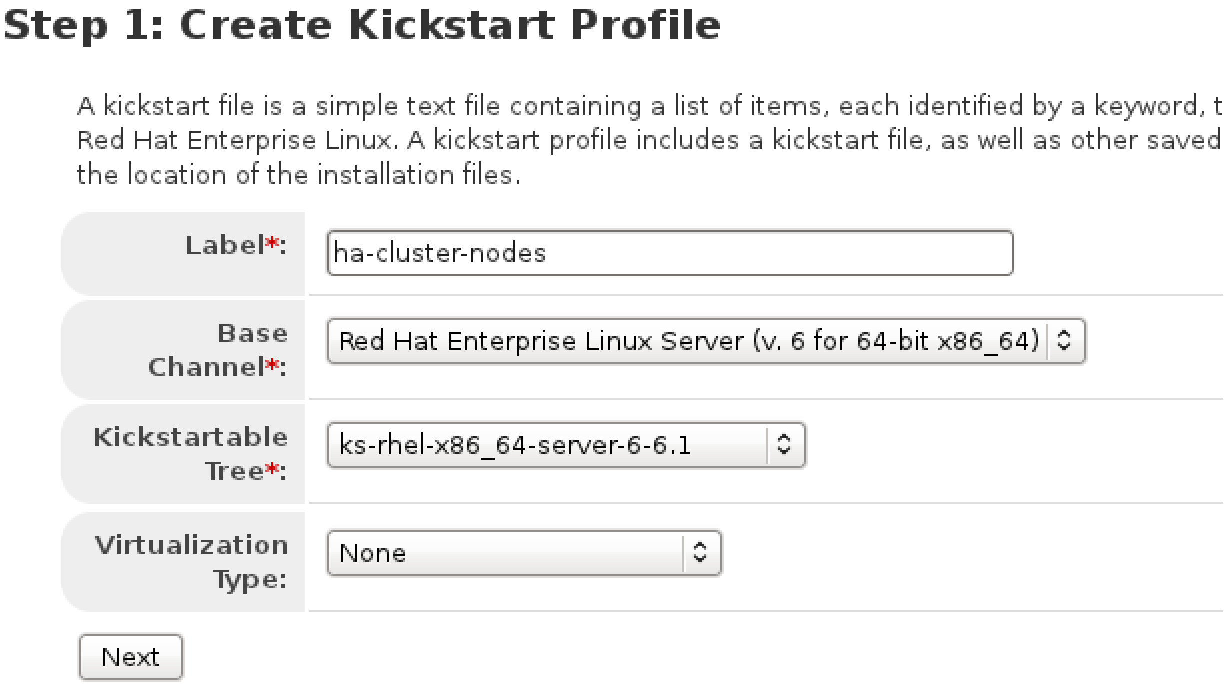
The following child channels of **Red Hat Enterprise Linux Server (v. 6 for 64-bit x86_64)**

- RHEL Server High Availability (v. 6 for 64-bit x86_64)
- RHEL Server Optional (v. 6 64-bit x86_64)
- RHEL Server Resilient Storage (v. 6 for 64-bit x86_64)
- RHEL Server Supplementary (v. 6 64-bit x86_64)
- RHEL V2VWIN (v. 6 for 64-bit x86_64)
- RHN Tools for RHEL (v. 6 for 64-bit x86_64)

Illustration 4.1-2: Child Channels

3. Using RHN Satellite, create a kickstart profile for the cluster nodes

   a) **Systems | Kickstart | Create New Kickstart Profile** and fill out Step 1. as shown in **Illustration 4.1-3: Create Kickstart Profile**



Step 1: Create Kickstart Profile

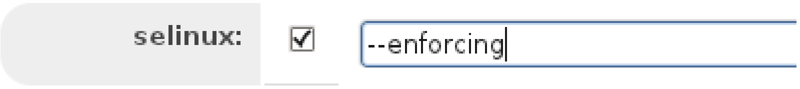
A kickstart file is a simple text file containing a list of items, each identified by a keyword, t
Red Hat Enterprise Linux. A kickstart profile includes a kickstart file, as well as other saved
the location of the installation files.

Label*:	ha-cluster-nodes
Base Channel*:	Red Hat Enterprise Linux Server (v. 6 for 64-bit x86_64)
Kickstartable Tree*:	ks-rhel-x86_64-server-6-6.1
Virtualization Type:	None

Illustration 4.1-3: Create Kickstart Profile

2. Select the defaults for Step 2
3. Provide a root password for Step 3. and click *Finish*

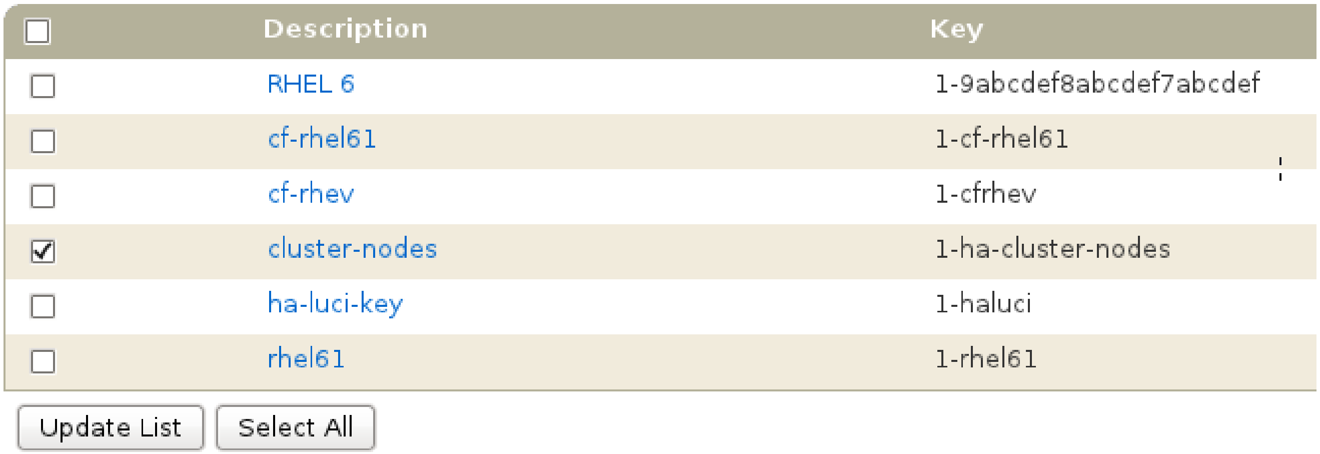
4. Enable SELinux in the kickstart profile, select the kickstart profile ha-cluster-nodes | Kickstart Details | Advanced Options and enable SELinux as shown in **Illustration 4.1-4: Enable SELinux**



selinux: ☑ --enforcing

Illustration 4.1-4: Enable SELinux

5. Click ***Update Kickstart***
6. Associate the activation key to the kickstart profile
   1. **Systems | Kickstart | View a List of Kickstart Profiles**
   2. Browse to the ha-cluster-nodes kickstart profile and associate the activation key by clicking the ***Activation Key*** link in the upper navigation menu and select the cluster -nodes activation key as shown in **Illustration 4.1-5: Associate Activation Key**



□	Description	Key
□	RHEL 6	1-9abcdef8abcdef7abcdef
□	cf-rhel61	1-cf-rhel61
□	cf-rhev	1-cfrhev
☑	cluster-nodes	1-ha-cluster-nodes
□	ha-luci-key	1-haluci
□	rhel61	1-rhel61

Illustration 4.1-5: Associate Activation Key

   3. Click *Update Activation Keys*

Deploy the Cluster Nodes

This reference architecture uses the Pre-Execution Environment (PXE) to deploy the cluster nodes. This is the quickest and easiest way but the nodes can be deployed by whatever methods are available.

1. PXE boot the cluster nodes and let them install
2. Make sure the right repositories were added

```
# yum repolist | grep High Availability
rhel-x86_64-server-ha-6 RHEL Server High Availability(v.6 for 64-bit x86_64)
```

3. Update the system when it is booted

```
# yum update
```

Configure Bonding for the Network Interfaces on Both Nodes

This configuration is not using a public / private network scenario. However, it is using bonding on the on-board Broadcom Ethernet interfaces. Please refer to the Red Hat knowledge-base article[7] for more information as the way bonding is implemented has changed. Perform these actions on both HA1 and HA2.

1. Create a *bonding.conf* file

```
# echo "alias bond0 bonding" >> /etc/modprobe.d/bonding.conf
```

2. Create the bonding interface file

```
# cat > /etc/sysconfig/network-scripts/ifcfg-bond0 <<EOF
DEVICE=bond0
IPADDR=10.16.139.40
NETMASK=255.255.248.0
USERCTL=no
BOOTPROTO=static
ONBOOT=yes
BONDING_OPTS="mode=1"
EOF
```

3. Configure the Ethernet interface file

Before on */etc/sysconfig/network-scripts/em1*[8] interface on HA1:

```
DEVICE=em1
HWADDR=F0:4D:A2:3B:A0:59
ONBOOT=yes
BOOTPROTO=dhcp
```

Before on */etc/sysconfig/network-scripts/em2* interface on HA1:

```
DEVICE=em2
BOOTPROTO=dhcp
HWADDR=f0:4d:a2:3b:a0:5b
ONBOOT=yes
```

After on */etc/sysconfig/network-scripts/em1* interface on HA1:

```
DEVICE=em1
BOOTPROTO=none
HWADDR=F0:4D:A2:3B:A0:59
ONBOOT=yes
MASTER=bond0
SLAVE=yes
USERCTL=no
```

After on */etc/sysconfig/network-scripts/em2* interface on HA1:

```
DEVICE=em2
BOOTPROTO=none
HWADDR=f0:4d:a2:3b:a0:5b
ONBOOT=yes
```

```
MASTER=bond0
SLAVE=yes
USERCTL=no
```

4. Restart networking

```
# service network restart
```

5. Confirm that *bond0* is up and running

```
# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.6.0 (September 26, 2009)

Bonding Mode: fault-tolerance (active-backup)
Primary Slave: None
Currently Active Slave: em1
MII Status: up
MII Polling Interval (ms): 0
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: em1
MII Status: up
Link Failure Count: 0
Permanent HW addr: f0:4d:a2:3b:af:41
Slave queue ID: 0

Slave Interface: em2
MII Status: up
Link Failure Count: 0
Permanent HW addr: f0:4d:a2:3b:af:43
Slave queue ID: 0
```

4.1.1 Secure the Cluster Nodes

This section discusses some options for securing the cluster nodes with *iptables.* Please refer to the proper Red Hat security documentation[9] and the *iptables* man page for more details. These systems require at least two types of traffic for the software to function properly. The first is for the cluster daemons to be able to communicate properly among each other. The second is for the NFS clients to be able to access the NFS share running in the cluster. Per the Red Hat knowledgebase[10] article NFS traffic will be locked down to certain ports. See for a complete copy of */etc/sysconfig/nfs.* Enforce the following rules on both HA1 and HA2.

Create a Custom Rule

1. Backup the current **iptables** configuration

```
# cp /etc/sysconfig/iptables{,.orig}
```

2. Display the current **iptables** configuration

```
# iptables –numeric –verbose --list --line-numbers
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out source destination
1      155 11804 ACCEPT     all  --  *      *  0.0.0.0/0 0.0.0.0/0
state RELATED,ESTABLISHED
```

```
2        0     0 ACCEPT     icmp --  *      *  0.0.0.0/0 0.0.0.0/0
3        0     0 ACCEPT     all  --  lo     *  0.0.0.0/0 0.0.0.0/0
4        0     0 ACCEPT     tcp  --  *      *  0.0.0.0/0 0.0.0.0/0
state NEW tcp dpt:22
5        8  3771 REJECT     all  --  *      * 0.0.0.0/0 0.0.0.0/0
reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source destination
1        0     0 REJECT     all  --  *      *  0.0.0.0/0   0.0.0.0/0
reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT 82 packets, 9576 bytes)
num   pkts bytes target     prot opt in     out     source destination
```

3. Create two new chains and forward the traffic to them

```
# iptables --new-chain ha-cluster
# iptables --new-chain ha-nfs
# iptables --insert INPUT --jump ha-nfs
# iptables --insert INPUT --jump ha-cluster
```

4. Add the cluster specific rules to the ha-cluster chain

```
# iptables --append ha-cluster --proto udp --destination-port 5404:5405
--jump ACCEPT
# iptables --append ha-cluster --proto tcp --dport 21064 --jump ACCEPT
# iptables --append ha-cluster --proto tcp --dport 16851 --jump ACCEPT
# iptables --append ha-cluster --proto tcp --dport 11111 --jump ACCEPT
# iptables --append ha-cluster --proto udp --dport 11111 --jump ACCEPT
```

5. Add the NFS specific rules to the ha-nfs chain

```
# iptables --append ha-nfs --proto tcp --dport 888 --jump ACCEPT
# iptables –-append ha-nfs --proto tcp --dport 889 --jump ACCEPT
# iptables --append ha-nfs --proto tcp --dport 890 --jump ACCEPT
# iptables --append ha-nfs --proto tcp --dport 891 --jump ACCEPT
# iptables --append ha-nfs --proto udp --dport 891 --jump ACCEPT
# iptables --append ha-nfs --proto udp --dport 890 --jump ACCEPT
# iptables --append ha-nfs --proto udp --dport 889 --jump ACCEPT
# iptables --append ha-nfs --proto udp --dport 888 --jump ACCEPT
# iptables --append ha-nfs --proto tcp --dport 2049 --jump ACCEPT
# iptables --append ha-nfs --proto udp --dport 2049 --jump ACCEPT
# iptables --append ha-nfs --proto tcp --dport 33179 --jump ACCEPT
# iptables --append ha-nfs --proto udp --dport 111 --jump ACCEPT
# iptables --append ha-nfs --proto tcp --dport 111 --jump ACCEPT
```

6. Save the rules

```
# service iptables save
# chkconfig iptables on
```

4.1.2 Configure Storage

The iSCSI network is on a dedicated 10GB/s Ethernet network. Bring up a third interface on both nodes to ensure connectivity over this network.

1. Verify that the network configuration file for the ifcfg-p6p1 interface is configured

properly and make sure the iSCSI target is pingable from each node.

- Note: In this paper, the Dell servers are utilizing a new feature called *biosdevname* that was released in Red Hat Enterprise Linux 6.1. This knowledgebase[11] explains the feature.

```
DEVICE=p6p1
BOOTPROTO=static
HWADDR=00:10:18:7e:cc:c8
IPADDR=192.168.0.1
NETMASK=255.255.255.0
ONBOOT=yes
```

2. Bring the interface up

```
# ifup p6p1
```

3. Ping the iSCSI target

```
# ping -c 1 192.168.0.20
PING 192.168.0.20 (192.168.0.20) 56(84) bytes of data.
64 bytes from 192.168.0.20: icmp_seq=1 ttl=255 time=0.053 ms

--- 192.168.0.20 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.053/0.053/0.053/0.000 ms
```

Discover and connect to the iSCSI target

Location: HA1 and HA2

Some storage arrays may use CHAP or LUN masking. The prerequisite here is that the storage array allows access to the iSCSI target via IP address or a iSCSI IQN of the initiator. Please refer to the storage vendors documentation for more information on customizing the storage.

1. Install the iscsi initator

```
# yum install iscsi-initiator-utils.x86_64
```

2. Set up CHAP on the Array and the cluster hosts. Below is an example */etc/iscsi/iscsid.conf*

```

[ ... output abbreviated ... ]

# *************
# CHAP Settings
# *************

# To enable CHAP authentication set node.session.auth.authmethod
# to CHAP. The default is None.
# Set auth type - scollier
node.session.auth.authmethod = CHAP

# To set a CHAP username and password for initiator
# authentication by the target(s), uncomment the following lines:
node.session.auth.username = iqn.1994-05.com.redhat:e0f140f2bb22
node.session.auth.password = <password>

# To set a CHAP username and password for target(s)
# authentication by the initiator, uncomment the following lines:
node.session.auth.username_in = reverseAuthentication
node.session.auth.password_in = <password>

# To enable CHAP authentication for a discovery session to the target
# set discovery.sendtargets.auth.authmethod to CHAP. The default is None.
discovery.sendtargets.auth.authmethod = CHAP

# To set a discovery session CHAP username and password for the initiator
# authentication by the target(s), uncomment the following lines:
discovery.sendtargets.auth.username = iqn.1994-05.com.redhat:e0f140f2bb22
discovery.sendtargets.auth.password = <password>

# To set a discovery session CHAP username and password for target(s)
# authentication by the initiator, uncomment the following lines:
discovery.sendtargets.auth.username_in = reverseAuthentication
discovery.sendtargets.auth.password_in = <password>

[ ... output abbreviated ... ]
```

3. Discover the iSCSI target

```
# iscsiadm --mode discovery --type sendtargets --portal 192.168.0.20
Starting iscsid:                                           [  OK  ]
192.168.0.20:3260,1 iqn.1992-05.com.emc:apm001043016750000-7
192.168.0.21:3260,1 iqn.1992-05.com.emc:apm001043016750000-7
```

3. Log in to the target

```
# iscsiadm --mode node --targetname \
iqn.1992-05.com.emc:apm001043016750000-7 --portal 192.168.0.21 –login
```

4. List the nodes

```
# iscsiadm --mode node
192.168.0.20:3260,1 iqn.1992-05.com.emc:apm001043016750000-7
192.168.0.21:3260,1 iqn.1992-05.com.emc:apm001043016750000-7
```

5. Reboot the server if necessary or rescan the SCSI bus. The command **rescan-scsi-bus.sh** comes from the sg3_utils package.

```
# rescan-scsi-bus.sh
```

6. List the device

```
# parted --list
```

7. Get the SCSI UUID off of the device. Perform this task on both nodes and make sure the UUID matches. This ID will be used later to configure **DM-Multipath** which will enable multiple I/O paths between server nodes and storage arrays into a single device.

```
# scsi_id --whitelisted --replace-whitespace --device=/dev/sdc
36006048ccdaa201f61defbc5685c82b1
```

Configure Multipath

1. Install the multipath software on both nodes

```
# yum install device-mapper-multipath.x86_64
```

2. To create a */etc/multipath.conf* file that disables *user friendly names* and starts multipathd

```
# mpathconf --enable --user_friendly_names n --with_multipathd y
```

3. Edit the */etc/multipath.conf* file and add the iSCSI device to create an alias from the previously obtained UUID. Refer to **Configuration Files**
4. Restart **multipathd** and check the paths and list the device in */dev/mapper*

```
# service multipathd reload
Reloading multipathd: [ OK ]

# multipath -ll
Jun 09 16:35:56 | multipath.conf line 31, invalid keyword: getuid_callout
Jun 09 16:35:56 | multipath.conf line 92, duplicate keyword: blacklist
EMCiSCSI (36006048ccdaa201f61defbc5685c82b1) dm-1 EMC,Celerra
size=223G features='0' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=1 status=active
| `- 11:0:0:0 sdb 8:16 active ready running
`-+- policy='round-robin 0' prio=1 status=enabled
  `- 12:0:0:0 sdc 8:32 active ready running

# ls /dev/mapper/
control  EMCiSCSI myvg-rootvol
```

Create a ext4 file system on EMCiSCSIp1

Location: HA1

1. Create the partition

```
# parted /dev/mapper/EMCiSCSI mkpart primary ext4 0 240GB
```

2. Create the ext4 file system

```
# mkfs.ext4 /dev/mapper/EMCiSCSIp1
```

or the XFS file system

```
# mkfs.xfs -d su=64k,sw=8 /dev/mapper/EMCiSCSIp1
```

4.2 Cluster Creation via Conga

During this phase the cluster is created. The following prerequisites have been met:

- Network Connectivity
- Shared Storage
- Cluster Management Software (Luci)
- Cluster Software (Ricci)
- Security

There are two ways to create a cluster with Red Hat Enterprise Linux 6. The first way, which is demonstrated in this section is via Conga. The second way, whichis demonstrated in the next section is via CCS.

4.2.1 Cluster Software Installation

At this point in the paper the management server, HA-LUCI, and cluster nodes, HA1 and HA2, have been installed and configured. All three systems: HA-LUCI, HA1, and HA2 have been secured with **iptables** and **SELinux** and the storage has been presented to HA1 and HA2. The next step in the process of deploying the cluster software is to install *ricci* on the cluster nodes HA1 and HA2. This section lists the process for HA1, the same process will need to be applied to HA2 for the cluster to be functional.

Install the cluster software on the cluster nodes

1. Install the High Availability group

```
# yum groupinstall "High Availability"
```

4.2.2 Cluster Creation

1. Set teh password for ricci on both of the cluster nodes

```
# passwd ricci
Changing password for user ricci.
New password: <password>
Retype new password: <password>
passwd: all authentication tokens updated successfully.
```

2. Start the *ricci* and *rgmanager* services on both of the cluster nodes

```
# service ricci restart

# chkconfig ricci on
```

3. From the *luci* web interface click ***Create*** and form a cluster as show in **Illustration 4.2.2-1: Create Cluster**

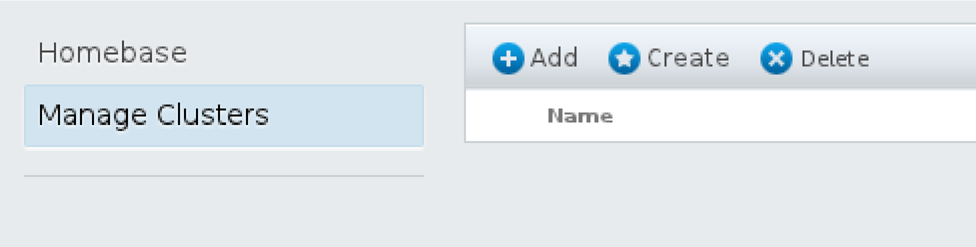
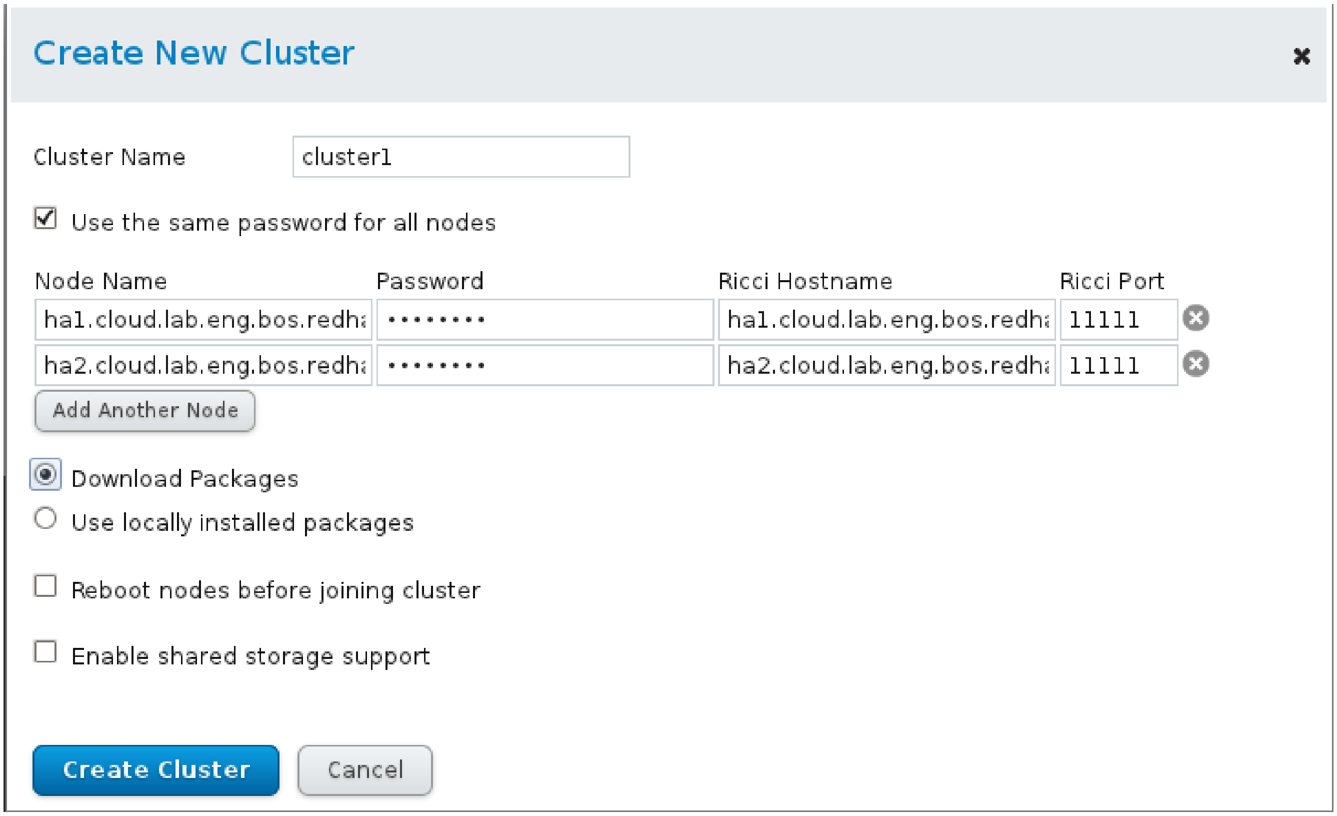


Illustration 4.2.2-1: Create Cluster

On the following screen, click ***Create*** and provide a name, use the same password as used for the *ricci* account on the cluster nodes, add the nodes and select ***Download Packages***. Then select ***Create Cluster*** as shown in **Illustration 4.2.2-2:Create Cluster**



Create New Cluster

Cluster Name: cluster1

☑ Use the same password for all nodes

Node Name	Password	Ricci Hostname	Ricci Port
ha1.cloud.lab.eng.bos.redh:	••••••••	ha1.cloud.lab.eng.bos.redh:	11111
ha2.cloud.lab.eng.bos.redh:	••••••••	ha2.cloud.lab.eng.bos.redh:	11111

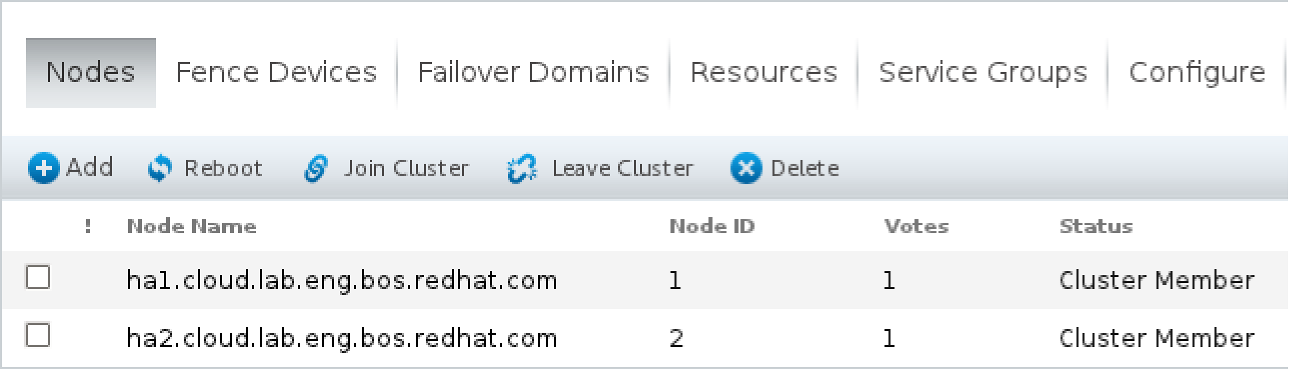
● Download Packages
○ Use locally installed packages

□ Reboot nodes before joining cluster

□ Enable shared storage support

Illustration 4.2.2-2:Create Cluster

4. After the nodes are installed they appear in the *luci* interface as in **Illustration 4.2.2-3: Cluster Nodes**



	!	Node Name	Node ID	Votes	Status
□		ha1.cloud.lab.eng.bos.redhat.com	1	1	Cluster Member
□		ha2.cloud.lab.eng.bos.redhat.com	2	1	Cluster Member

Illustration 4.2.2-3: Cluster Nodes

4.2.3 Fencing

This reference architecture utilizes two methods for fencing. The primary method for fencing is with a network capable power distribution unit. The second is with IPMI over LAN which connects to the Dell DRAC 6. Both of these methods provide power control to the servers which can help prevent data corruption if there are problems with the cluster. Ensure that the IPMI interfaces on the servers and the network based power distribution units can be reached from the cluster Ethernet interfaces on both nodes.

For network power switch (NPS) fencing, add one fence device per NPS, in this case APC. After the devices are added, add multiple instances to the fence method depending on how many power outlets are used. Tables **Table 4.2.3-1: APC 1 Configuration** and **Table 4.2.3-2: APC 2 Configuration** show the APC port / IP setup for this reference architecture.

APC 1	Parameters
Username	refarch
Password	**<password>**
IP Address	10.16.128.45
SSH Protocol	v1[12]
HA1 Power Supply 1	Port 20
HA2 Power Supply 1	Port 22

Table 4.2.3-1: APC 1 Configuration

APC2	Parameters
Username	refarch
Password	**<password>**
IP Address	10.16.128.46
SSH Protocol	v1
HA1 Power Supply 2	Port 5
HA2 Power Supply 2	Port 3

Table 4.2.3-2: APC 2 Configuration

Configuring a Network Power Switch Fencing Device

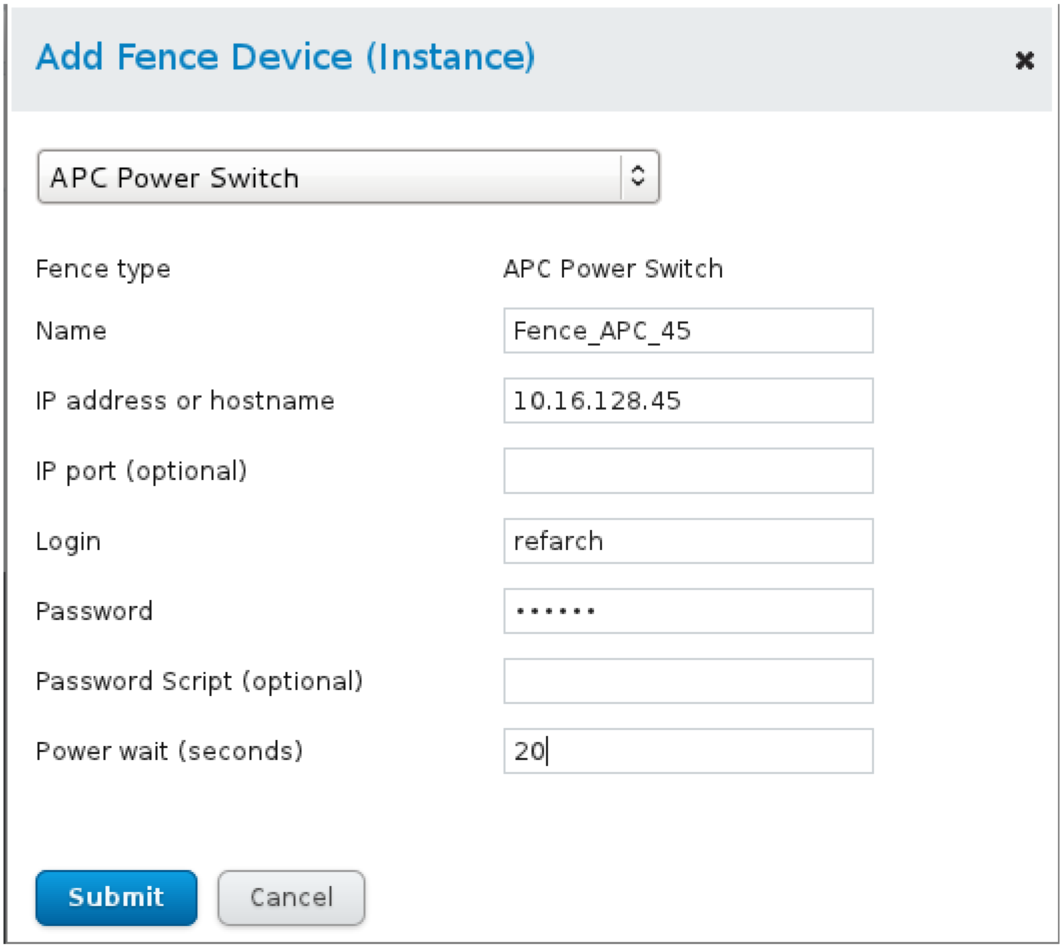
Following are the steps that provide a high level overview of configuring the NPS fencing device:

1. Create the fence device instances for both cluster nodes
2. Create the fence methods for each cluster nodes
3. Associate the fence devices to the appropriate fence methods

Detailed steps follow below.

Location: Browser connected to *luci* interface

1. Add the fence devices, click on **Fence Devices** in the upper navigation menu, click **Add** and choose the **APC Power Switch** from the drop-down menu as shown in **Illustration 4.2.3-1: Configure APC 1 Fence Device** and click **Submit**. The **power wait** option dictates how long to wait after issuing a power off or a power on command.



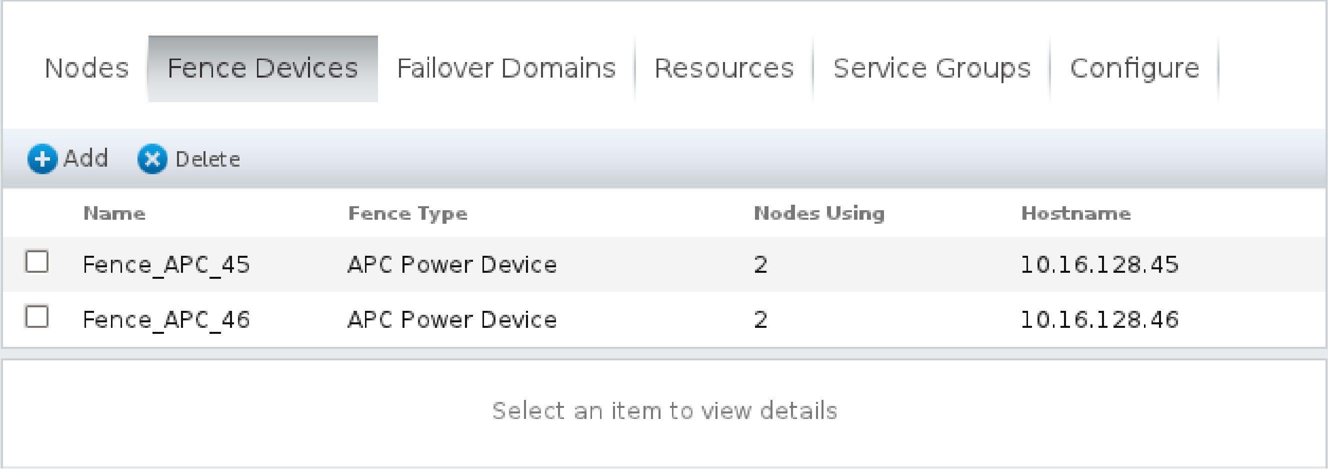
Add Fence Device (Instance)

APC Power Switch

Fence type	APC Power Switch
Name	Fence_APC_45
IP address or hostname	10.16.128.45
IP port (optional)	
Login	refarch
Password	••••••
Password Script (optional)	
Power wait (seconds)	20

Illustration 4.2.3-1: Configure APC 1 Fence Device

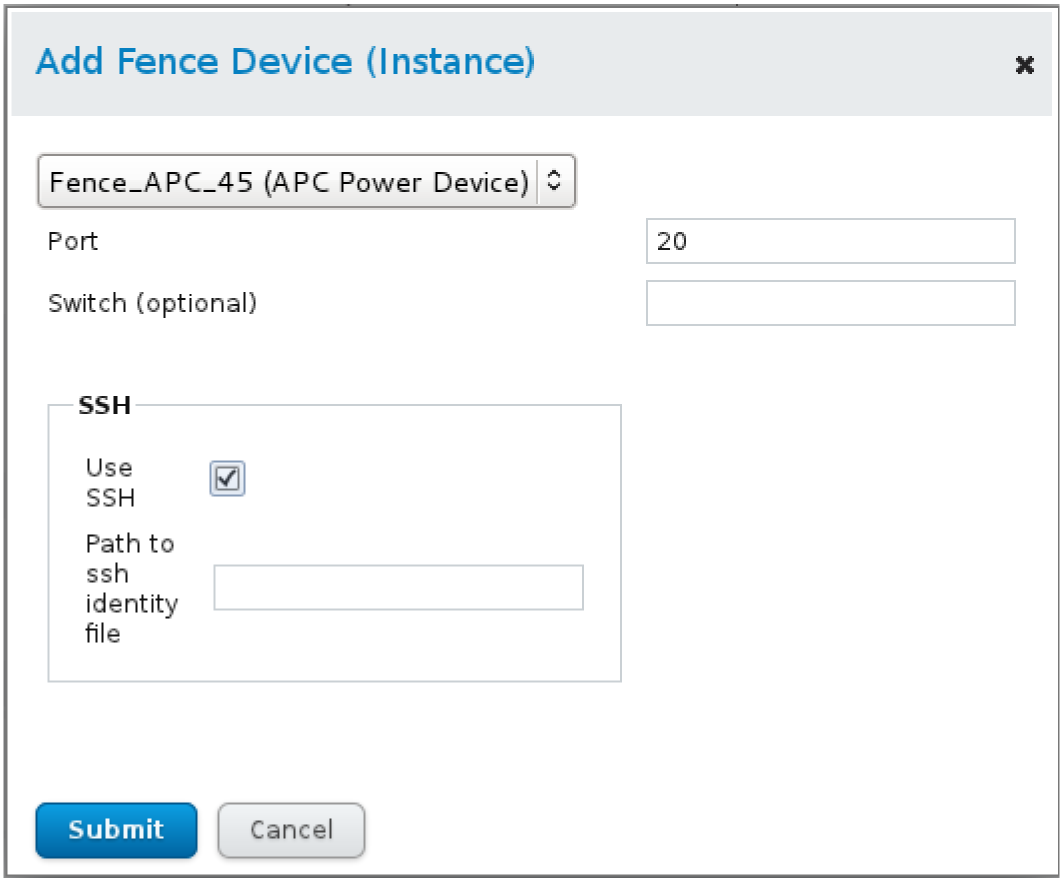
Add a second device as shown in **Illustration 4.2.3-2: APC Fence Devices** with the specified parameters and click **Submit**.



	Name	Fence Type	Nodes Using	Hostname
□	Fence_APC_45	APC Power Device	2	10.16.128.45
□	Fence_APC_46	APC Power Device	2	10.16.128.46

Select an item to view details

Illustration 4.2.3-2: APC Fence Devices



Add Fence Device (Instance)

Fence_APC_45 (APC Power Device)

Port: 20

Switch (optional):

SSH

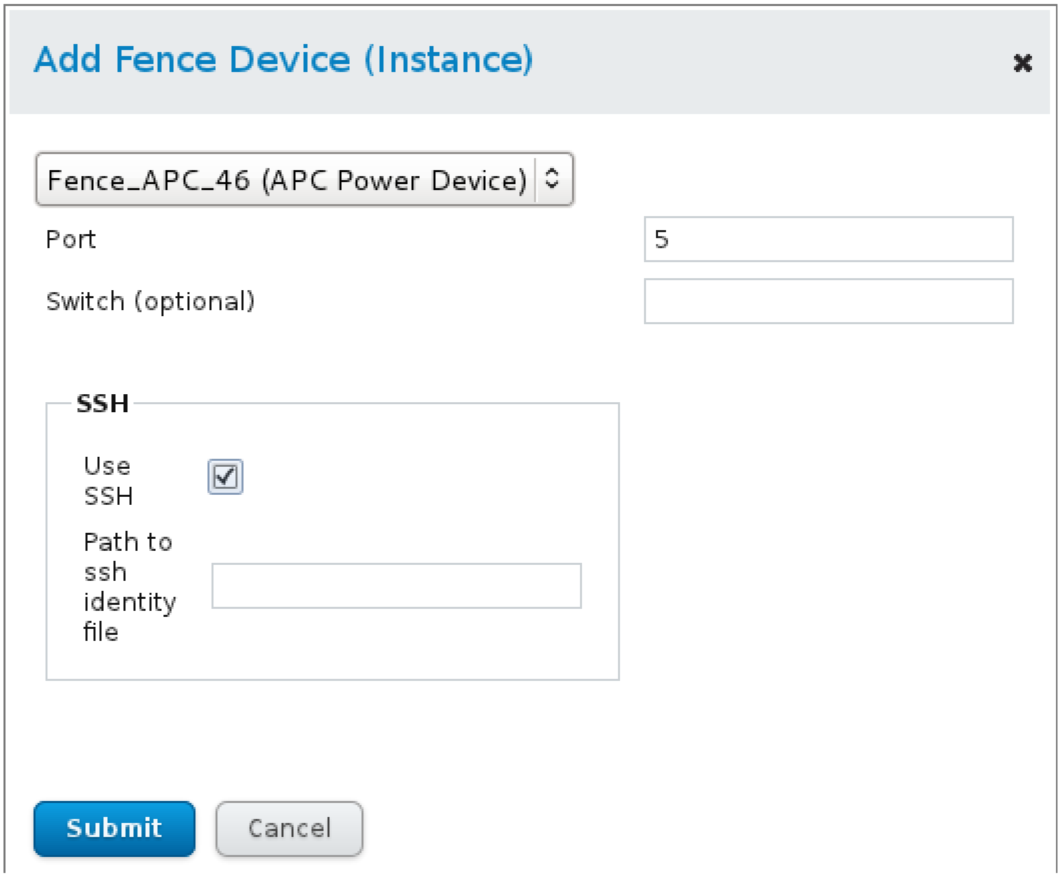
Use SSH ☑

Path to ssh identity file:

Illustration 4.2.3-3: HA1 Device to Method Association Port 20

2. Associate the fence devices to the nodes by clicking on ***Nodes*** in the upper navigation menu, select HA1, click ***Add Fence Method*** and provide a fence method name of HA1_Method_APC and click Submit. Once the fence method is added click the ***Add Fence Instance****,* select Fence_APC_45, provide the parameters as shown in **Illustration 4.2.3-3: HA1 Device to Method Association Port 20** and click ***Submit.***

3. Add the device to the HA1_Method_APC method with the other port now as shown in **Illustration 4.2.3-4: HA1 Device to Method Association Port 5** by clicking on ***Add Fence Instance*** and choosing the Fence_APC_46 device. The ports signify the location of the power cables on the APC unit.



Add Fence Device (Instance)

Fence_APC_46 (APC Power Device)

Port: 5

Switch (optional):

SSH

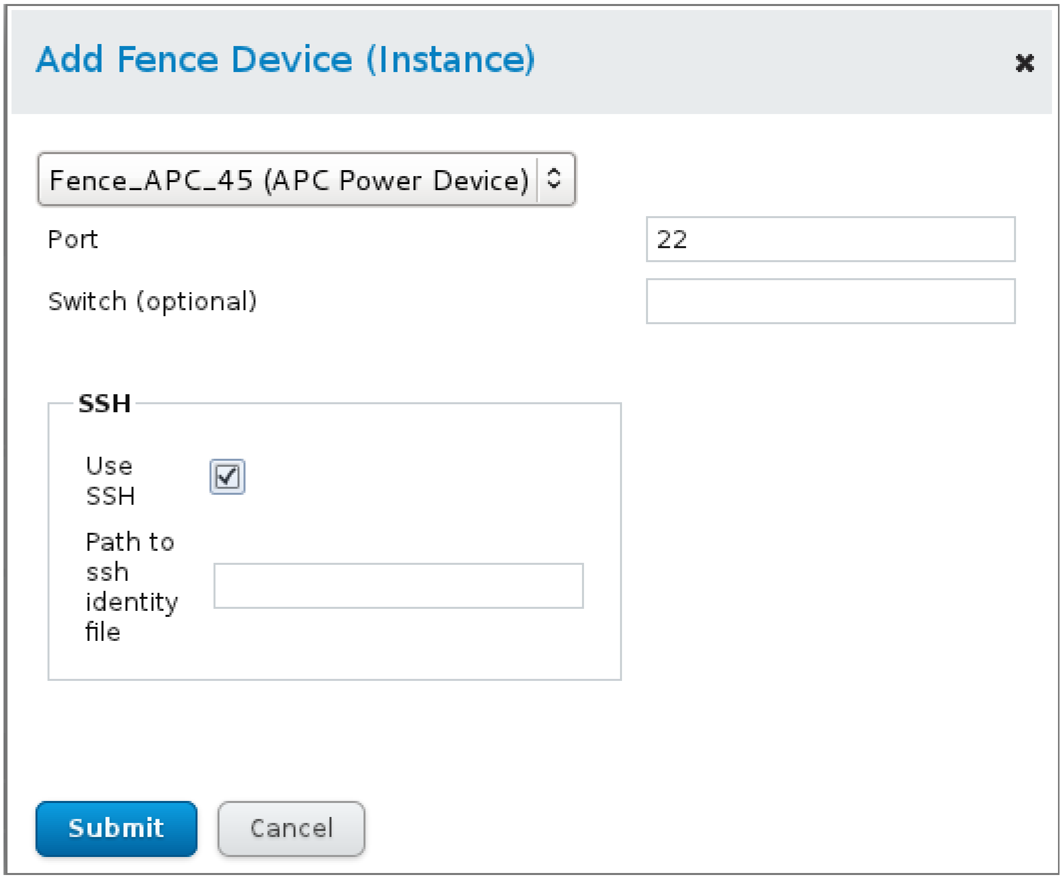
Use SSH ☑

Path to ssh identity file:

Illustration 4.2.3-4: HA1 Device to Method Association Port 5

4. Add a second Fence Method for HA2, associate the APC fence devices to this method. Choose the node HA2, click ***Add Fence Method*** and populate with the name HA2_Method_APC and click ***Submit.***

5. Click ***Add Fence Instance*** and add the following fence devices to the HA2_Method_APC as shown in **Illustration 4.2.3-5: HA2 Device to Method Association Port 22**



Add Fence Device (Instance)

Fence_APC_45 (APC Power Device)

Port: 22

Switch (optional):

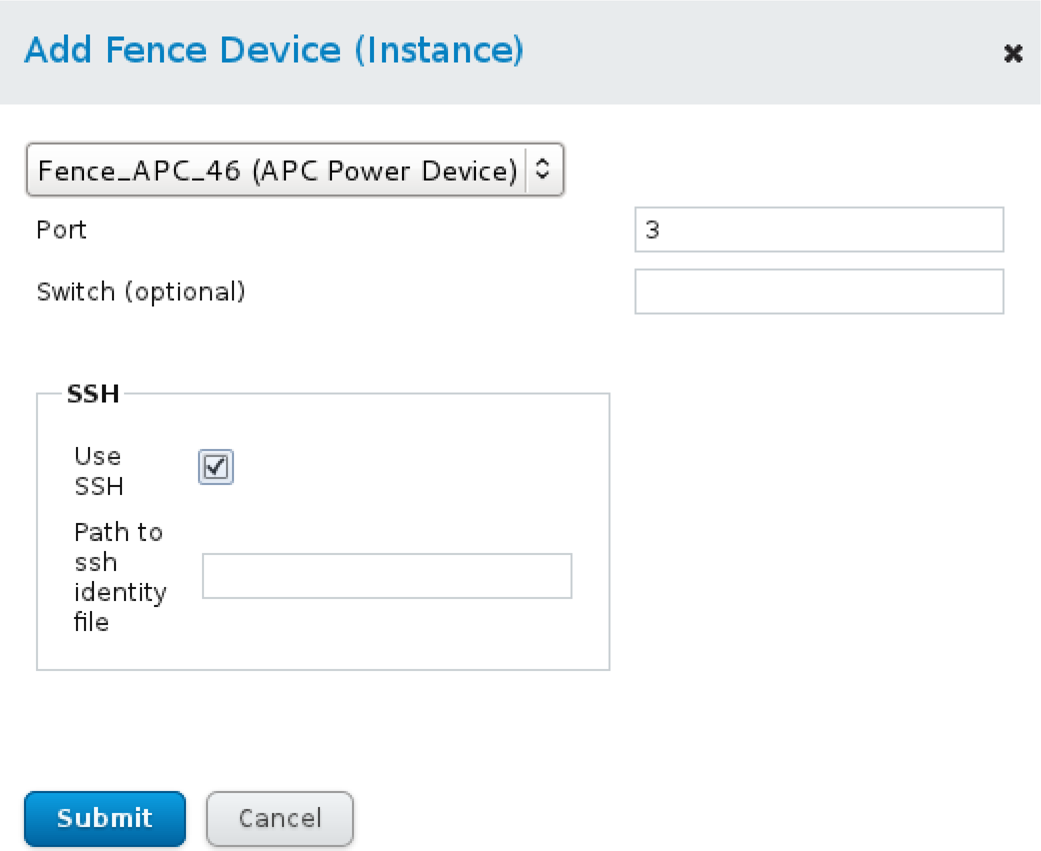
SSH

Use SSH ☑

Path to ssh identity file:

Illustration 4.2.3-5: HA2 Device to Method Association Port 22

6. Click on **Add Fence Instance** and add the second port as shown in **Illustration 4.2.3-6: HA2 Device to Method Association Port 3** and click ***Submit*.**



Add Fence Device (Instance)

Fence_APC_46 (APC Power Device)

Port: 3

Switch (optional):

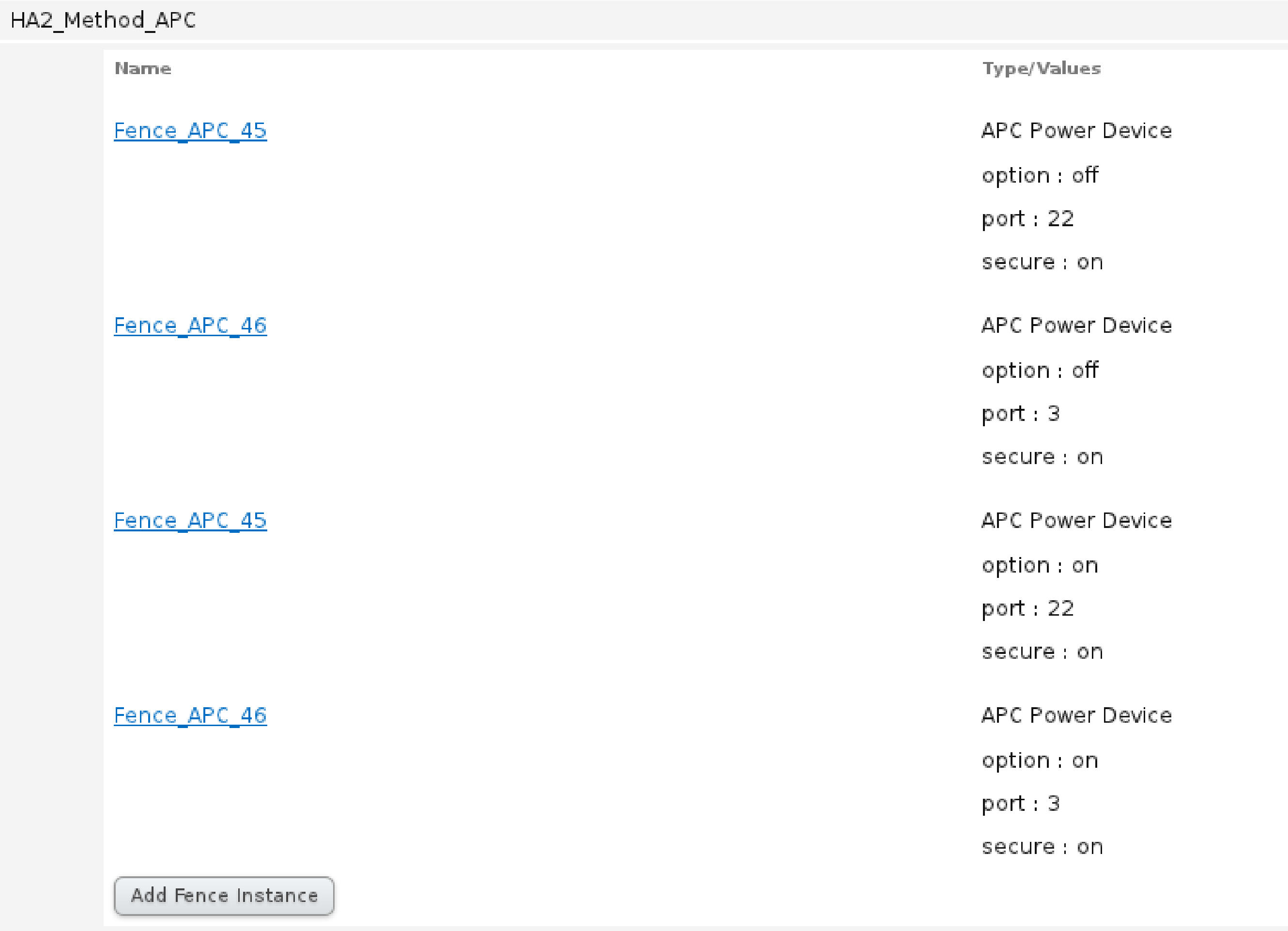
SSH

Use SSH ☑

Path to ssh identity file:

Illustration 4.2.3-6: HA2 Device to Method Association Port 3

7. Once the devices have been associated with the method for each server, the Fence Devices section should look like **Illustration 4.2.3-7: HA2 Fencing Configuration**



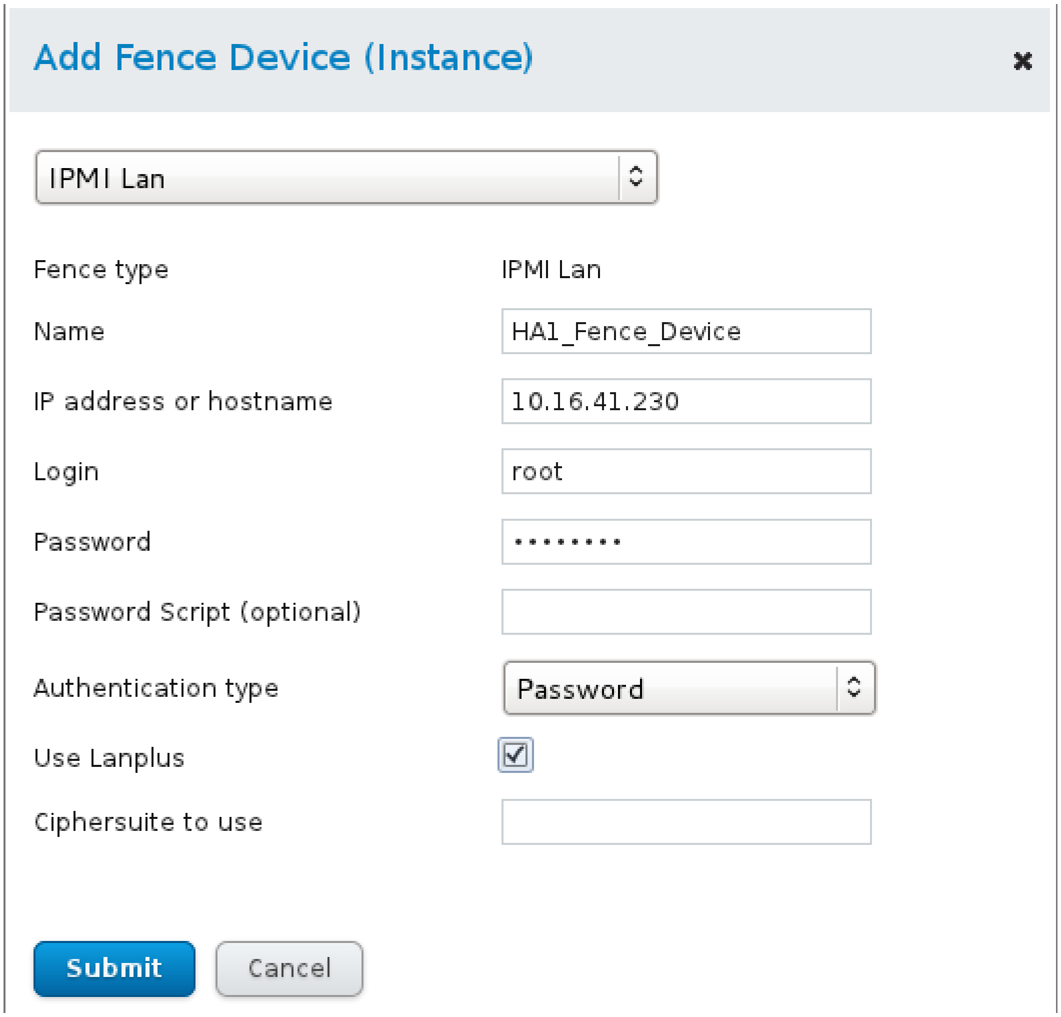
HA2_Method_APC

Name	Type/Values
Fence_APC_45	APC Power Device option : off port : 22 secure : on
Fence_APC_46	APC Power Device option : off port : 3 secure : on
Fence_APC_45	APC Power Device option : on port : 22 secure : on
Fence_APC_46	APC Power Device option : on port : 3 secure : on

Illustration 4.2.3-7: HA2 Fencing Configuration

Configure IPMI LAN Fencing Device

1. Click on the ***Fence Devices*** tab in the upper navigation menu to add a fence device and click ***Add.*** Provide the following information as shown in **Illustration 4.2.3-8: HA1 Fence Device** and then click ***Submit***

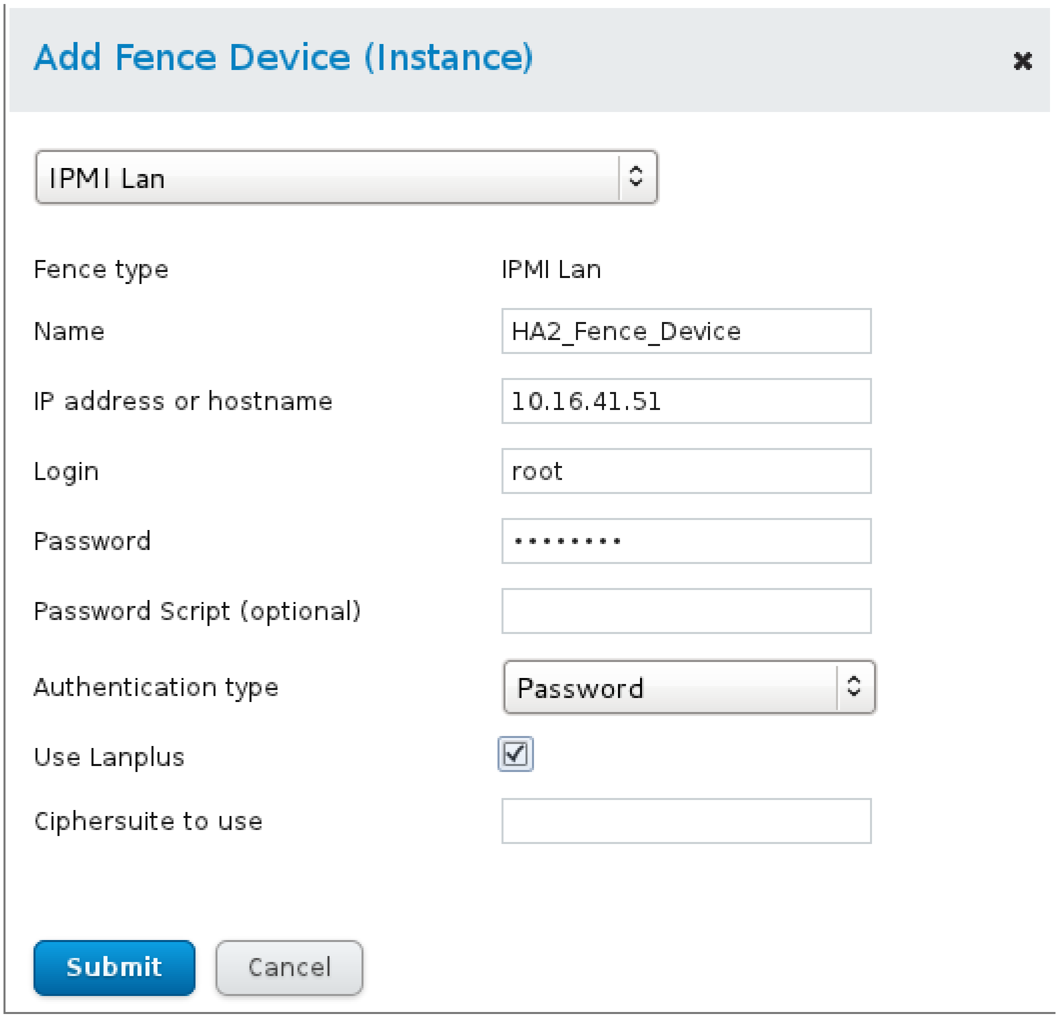


IPMI Lan

Fence type	IPMI Lan
Name	HA1_Fence_Device
IP address or hostname	10.16.41.230
Login	root
Password	•••••••••
Password Script (optional)	
Authentication type	Password
Use Lanplus	☑
Ciphersuite to use	

Illustration 4.2.3-8: HA1 Fence Device

2. Click on the ***Fence Devices*** tab in the upper navigation menu to add a fence device and click ***Add.*** Provide the following information as shown in **Illustration 4.2.3-9: HA2 Fence Device** and then click ***Submit***



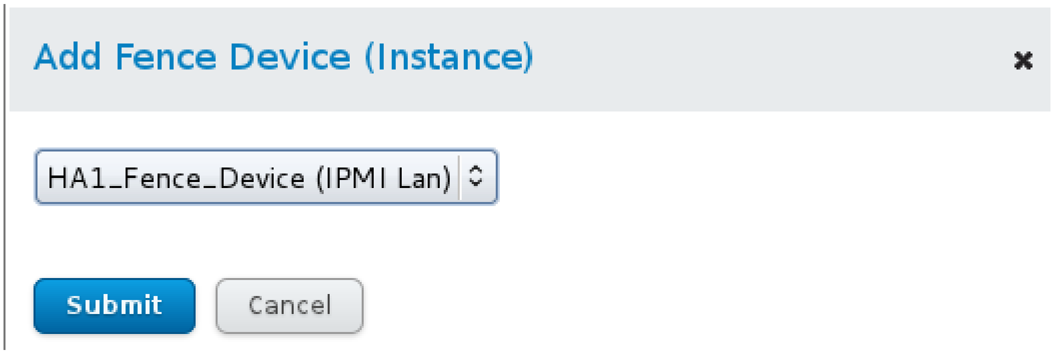
Add Fence Device (Instance)

IPMI Lan

Fence type	IPMI Lan
Name	HA2_Fence_Device
IP address or hostname	10.16.41.51
Login	root
Password	•••••••••
Password Script (optional)	
Authentication type	Password
Use Lanplus	☑
Ciphersuite to use	

Illustration 4.2.3-9: HA2 Fence Device

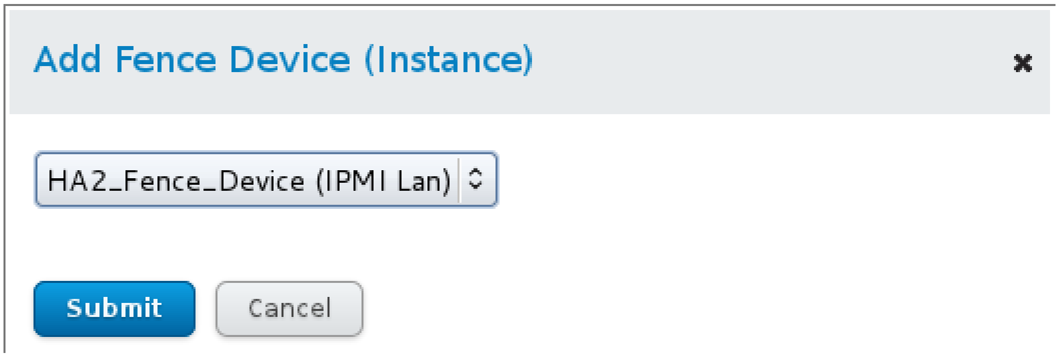
2. Associate the fence devices with the appropriate nodes
    1. Click on ***Nodes*** in the upper navigation menu and select HA1 and click ***Add Fence Method*** and providea name of HA1_Method, next add a ***Fence Instance***. Perform this for both servers as shown in **Illustration 4.2.3-10: HA1 Fence Instance** and **Illustration 4.2.3-11: HA2 Fence Instance**



Add Fence Device (Instance)

HA1_Fence_Device (IPMI Lan)

Illustration 4.2.3-10: HA1 Fence Instance



Add Fence Device (Instance)

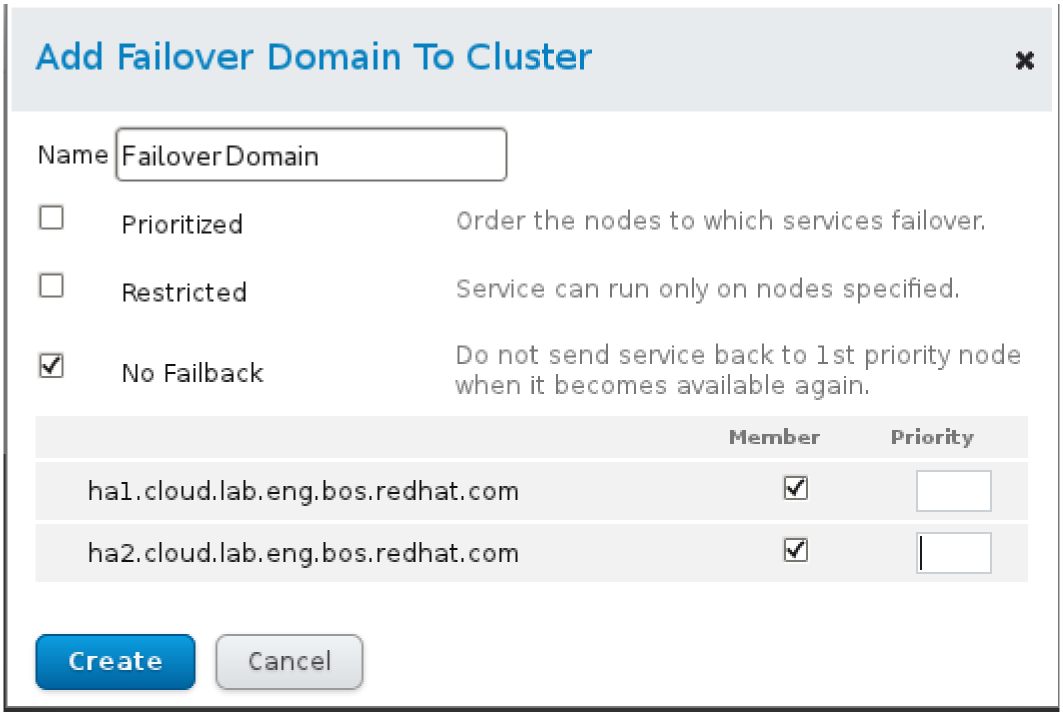
HA2_Fence_Device (IPMI Lan)

Illustration 4.2.3-11: HA2 Fence Instance

Once all the fence methods and devices have been added, the APC fencing method is the primary fence device and the IPMI LAN method is secondary fence device.

4.2.4 Failover Domains

Failover Domains are only necessary when prioritizing nodes in the cluster. Click on the ***Failover Domains*** tab in the upper navigation menu and then click ***Add***. Provide a ***Name*** and policy as shown below, select the members and click ***Create*. The *failover* domain properties should look like **Illustration 4.2.4-1: Create Failover Domain**



Add Failover Domain To Cluster ✖

Name	FailoverDomain	
□	Prioritized	Order the nodes to which services failover.
□	Restricted	Service can run only on nodes specified.
☑	No Failback	Do not send service back to 1st priority node when it becomes available again.

	Member	Priority
ha1.cloud.lab.eng.bos.redhat.com	☑	
ha2.cloud.lab.eng.bos.redhat.com	☑	

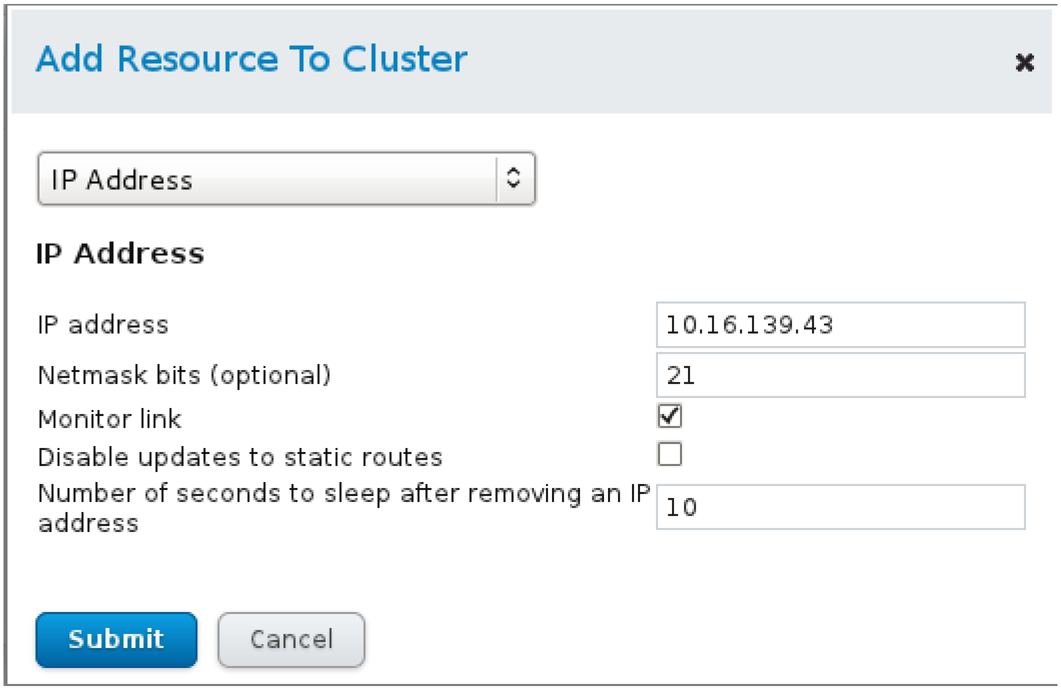
Illustration 4.2.4-1: Create Failover Domain

4.2.5 Cluster Resources

This section will list the resources that are configured for this cluster. Cluster resources are building blocks that you create and manage in the cluster configuration file - for example, an IP address, an application initialization script, or a Red Hat GFS2 shared partition. The resources used in this paper are:

- IP Address
- File System
- NFS Export
- NFS Client

1. Click ***Resources*** on the upper navigation menu and click ***Add***.
2. Select **IP Address** from the drop-down menu and fill out the options as shown in **Illustration 4.2.5-1: Adding IP Address** and click ***Submit***. For the Number of seconds to sleep after removing an IP address, it is safe to take the defaults when using NFS, if using something other than NFS, it is safe to set it to zero.



Add Resource To Cluster

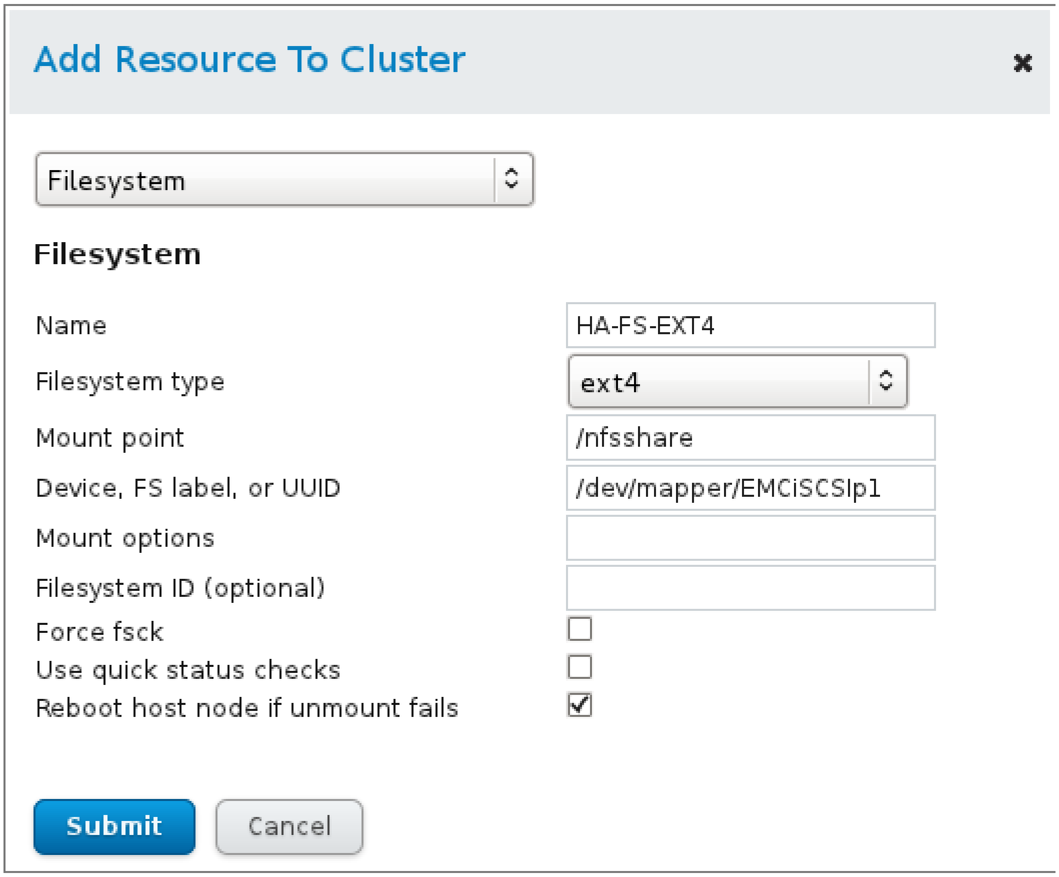
IP Address

IP Address

IP address	10.16.139.43
Netmask bits (optional)	21
Monitor link	☑
Disable updates to static routes	□
Number of seconds to sleep after removing an IP address	10

Illustration 4.2.5-1: Adding IP Address

3. Click ***Add*** and select file system from the drop-down menu as shown in **Illustration 4.2.5-2: Add File System to Cluster** and click ***Submit.*** If the file system is XFS, pick XFS from the **Filesystem type** dropdown.



Add Resource To Cluster

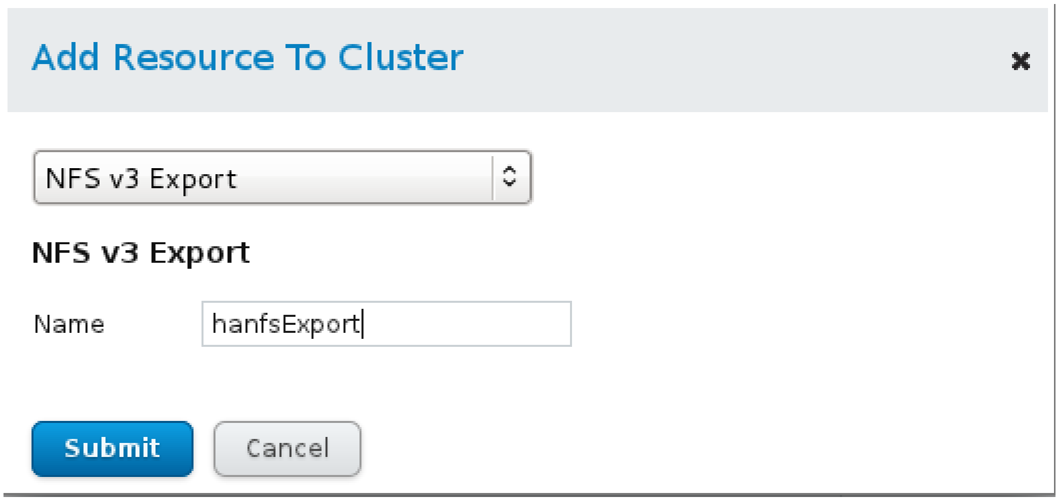
Filesystem

Filesystem

Name	HA-FS-EXT4
Filesystem type	ext4
Mount point	/nfsshare
Device, FS label, or UUID	/dev/mapper/EMCiSCSIp1
Mount options	
Filesystem ID (optional)	
Force fsck	□
Use quick status checks	□
Reboot host node if unmount fails	☑

Illustration 4.2.5-2: Add File System to Cluster

4. Click ***Add*** and select *NFS v3 Export* from the drop-down menu as shown in **Illustration 4.2.5-3: Adding NFS Export** and click ***Submit.***



NFS v3 Export

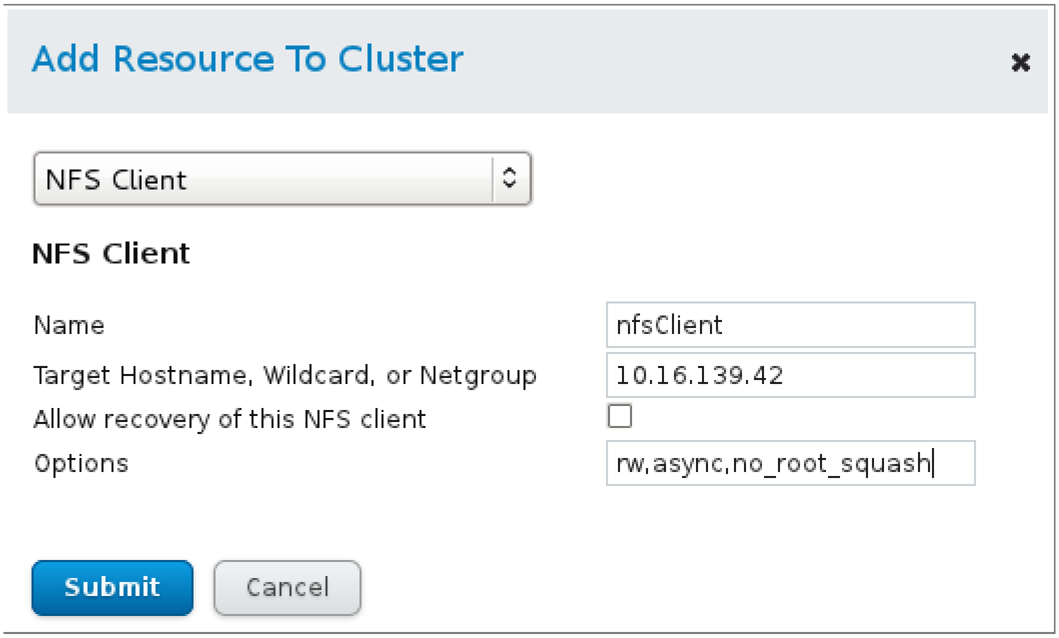
NFS v3 Export

Name: hanfsExport

Illustration 4.2.5-3: Adding NFS Export

5. Click ***Add*** and select ***NFS Client*** from the drop-down menu as shown in **Illustration 4.2.5-4: Adding NFS Client** and click ***Submit.*** The options provide the following functinality:

- *rw* - Allow both read and write access to the share
- *async* - replies to NFS client before the data is written to disk. This improves performance but could result in lost data if the node goes down.
- *no_root_squash* - allows root to connect as root and write to the NFS share



NFS Client

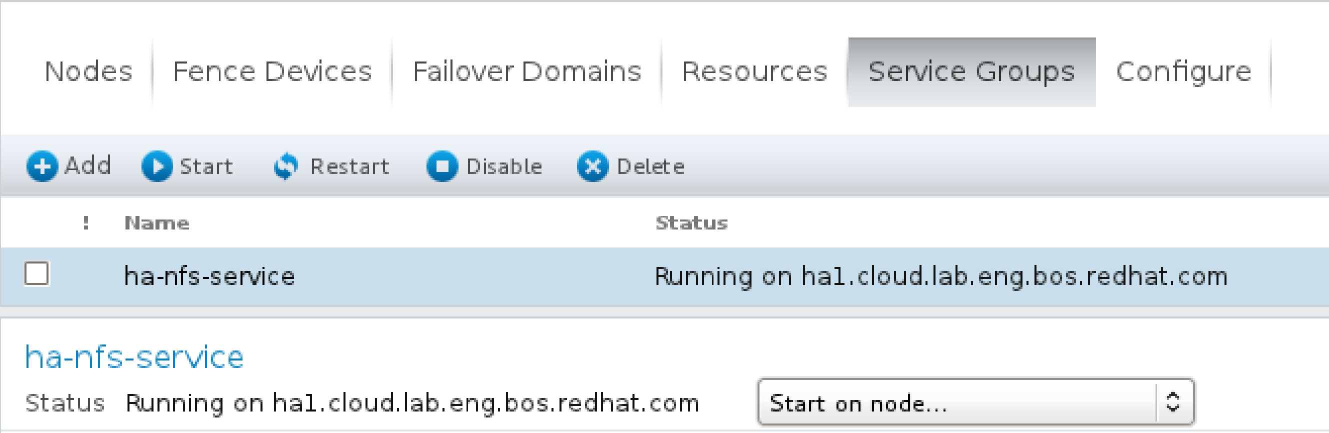
NFS Client

Name	nfsClient
Target Hostname, Wildcard, or Netgroup	10.16.139.42
Allow recovery of this NFS client	□
Options	rw,async,no_root_squash

Illustration 4.2.5-4: Adding NFS Client

4.2.6 Cluster Services

1. Click on ***Service Groups*** in the upper navigation menu and click ***Add.*** Fill out the properties of the service and click ***Submit.***
    1. Click ***Add Resource*** and select the IP Address resource from the drop-down menu
    2. Click ***Add a Child Resource*** and select ***File system*** Resource from the drop-down menu. This makes the File System a child of the IP Address.
    3. Click ***Add a Child Resource*** and select the *NFS Export* resource from the drop-down menu. This makes the *NFS Export* a child of the *File System*.
    4. Click ***Add a Child Resource*** and select the *NFS Client* resource from the drop-down menu and click ***Submit.*** This makes the *NFS Client* a child of the *NFS Export*.
    5. Start the resource from the ***Service Groups*** tab by checking the service and clicking the ***Start*** button as shown in **Illustration 4.2.6-1: Start NFS Resource**



	!	Name	Status
□		ha-nfs-service	Running on hal.cloud.lab.eng.bos.redhat.com

ha-nfs-service

Status Running on hal.cloud.lab.eng.bos.redhat.com

Illustration 4.2.6-1: Start NFS Resource

4.3 Cluster Creation via CCS

When using the CCS tool, there are two ways that the cluster settings can be created / modified. The first way is to edit an existing *cluster.conf* file on remote cluster nodes. The second is that CCS creates a cluster from scratch by creating a *cluster.conf* file locally that is pushed out to the cluster nodes. This section evaluates the second method and builds a cluster as well as the services from scratch and then push the *cluster.conf* file out to the cluster nodes. This section isperformed on cluster nodes that were cleanly re-installed.

There are a few actions that CCS can perform, the first being a query. CCS can query a remote host and pull information like version number, services, fence devices, etc. Second, CCS can modify a *cluster.conf* file. Finally, CCS can synchronize the cluster configuration throughout the cluster as well as start and stop cluster services.

This section provides a script which creates a local *cluster.conf* file and performs the following tasks:

1. Checks for existence of *cluster.conf* file and if it's there, back it up
2. Create the cluster infrastructure
3. Add primary fence methods, devices and instances
4. Add secondary fence methods, devices and instances
5. Add the failover domain
6. Add the resources
7. Add the service
8. Validate the configuration
9. Activate the configuration

4.3.1 Install the Cluster Software

Location: HA1 and HA2

1. Install the high availability group with yum

```
# yum groupinstall “High Availability”
```

2. Set the password for ricci and chkconfig the service on

```
# passwd ricci
# chkconfig ricci on
```

4.3.2 Run the CCS Script

The CCS script is called *cluster-config.sh* and can be run from root's home directory. This contents of the script can be modified to match the cluster environment. The contents below can be copied and pasted to create the script and then make the script executable and run it.

Location: HA1 and HA2

```
# service rgmanager start && service ricci start
# mkdir /nfsshare
```

Location: HA-LUCI

```
#!/bin/bash
# This script will create a highly available cluster provide NFS services

# Start script logging
(

# Set variables
MAINDIR="cluster_configuration"
CONFBACKUP="$MAINDIR/cluster_backup"
LOGSCRIPT="$MAINDIR/cluster_script_logs"
CCSGEN="ccs -f cluster.conf"
NODE1="ha1.cloud.lab.eng.bos.redhat.com"
NODE2="ha2.cloud.lab.eng.bos.redhat.com"
FSRESOURCE="/dev/mapper/EMCiSCSIp1"
IPRESOURCE="10.16.139.46/21"

# Create logging directories
mkdir -p $CONFBACKUP
mkdir -p $LOGSCRIPT

# Function to check if ccs exists
CCSEXISTS() {
if [ ! -f /usr/sbin/ccs ]; then
        echo
        echo "ccs is not installed"
        echo "make sure this system is registered with the HA rhn channel"
        echo
        exit 1
fi
}

# Function to test if the cluster.conf file exists
CLUSTERFILE() {
if [ -f cluster.conf ]; then
        echo
        echo "cluster.conf exists, backing up"
        echo
        cp cluster.conf $CONFBACKUP/cluster.conf-`date +%F_%T`.backup
fi
}

# Function to validate the cluster.conf file
VALIDATE() {
echo
echo "Validating the cluster.conf"
xmllint \-\-relaxng /usr/share/ccs/cluster.rng cluster.conf &> /dev/null

if [ $? == 0 ]; then
        echo "cluster.conf is validated"
else
```

```
        echo "cluster.conf is not validated"
fi
}

# Function to deploy cluster.conf to nodes and start clustering
DEPLOY() {
echo "Push out the configuration to $NODE1 and $NODE2"
$CCSGEN -h $NODE1 --setconf
$CCSGEN -h $NODE2 --setconf

echo "Synchronize and activate the cluster.conf file on the nodes $NODE1 and
$NODE2"
ccs -h $NODE1 --sync --activate
ccs -h $NODE2 --sync --activate

echo "Starting cluster services"
ccs -f cluster.conf --startall
}

# check for existence of ccs and cluster.conf file
CCSEXISTS
CLUSTERFILE

echo
echo "This script is creating the cluster from scratch, on a local system
that will be copied to the cluster nodes"
$CCSGEN --createcluster ccsCluster

# add the nodes
$CCSGEN --addnode $NODE1
$CCSGEN --addnode $NODE2

# BEGIN ADDING APC FENCING
echo
echo "Start of adding APC Fence Configuration"
$CCSGEN --addmethod HA1_Method_APC $NODE1
$CCSGEN --addmethod HA2_Method_APC $NODE2

echo "Adding APC fence devices"
$CCSGEN --addfencedev Fence_APC_45 agent="fence_apc" ipaddr="10.16.128.45"
login="refarch" passwd="a22BCv" power_wait="20"
$CCSGEN --addfencedev Fence_APC_46 agent="fence_apc" ipaddr="10.16.128.46"
login="refarch" passwd="a22BCv" power_wait="20"

echo "Adding APC fence instances"
$CCSGEN --addfenceinst Fence_APC_45 $NODE1 HA1_Method_APC option="off"
port="20" secure="on"
$CCSGEN --addfenceinst Fence_APC_46 $NODE1 HA1_Method_APC option="off"
port="5" secure="on"
$CCSGEN --addfenceinst Fence_APC_45 $NODE1 HA1_Method_APC option="on"
port="20" secure="on"
$CCSGEN --addfenceinst Fence_APC_46 $NODE1 HA1_Method_APC option="on"
port="5" secure="on"

$CCSGEN --addfenceinst Fence_APC_45 $NODE2 HA2_Method_APC option="off"
```

```
port="22" secure="on"
$CCSGEN --addfenceinst Fence_APC_46 $NODE2 HA2_Method_APC option="off"
port="3" secure="on"
$CCSGEN --addfenceinst Fence_APC_45 $NODE2 HA2_Method_APC option="on"
port="22" secure="on"
$CCSGEN --addfenceinst Fence_APC_46 $NODE2 HA2_Method_APC option="on"
port="3" secure="on"
echo "End of adding APC Fence Configuration"
echo
# END APC FENCING

# BEGIN IPMI LAN FENCING
# add fence method
echo "Start of adding IPMI LAN Fence Configuration"
$CCSGEN --addmethod fence_ipmilan $NODE1
$CCSGEN --addmethod fence_ipmilan $NODE2

echo "Adding IMPI LAN fence devices"
$CCSGEN --addfencedev ha1_fence agent=fence_ipmilan auth=password
ipaddr=10.16.41.230 lanplus=on login=root name=ha1_fence passwd=<password>
$CCSGEN --addfencedev ha2_fence agent=fence_ipmilan auth=password
ipaddr=10.16.41.51 lanplus=on login=root name=ha1_fence passwd=<password>

echo "Add IMPI LAN fence instances"
$CCSGEN --addfenceinst ha1_fence $NODE1 fence_ipmilan
$CCSGEN --addfenceinst ha2_fence $NODE2 fence_ipmilan
echo "End of adding IPMI LAN Fence Configuration"
echo
# END IMPI LAN FENCING

# BEGIN FAILOVER DOMAIN
echo "Add failover domain"
$CCSGEN --addfailoverdomain ccsFailoverDomain

echo "Add nodes to failover domain"
$CCSGEN  --addfailoverdomainnode ccsFailoverDomain $NODE1
$CCSGEN  --addfailoverdomainnode ccsFailoverDomain $NODE2
# END FAILOVER DOMAIN

# BEGIN ADDING RESOURCES
echo "Add IP Address resource"
$CCSGEN --addresource ip address="$IPRESOURCE" sleeptime=10
echo "Add File system resource"
$CCSGEN --addresource fs device="$FSRESOURCE" mountpoint="/nfsshare"
name="fileSystem"

echo "Add NFS Export"
$CCSGEN --addresource nfsexport name="nfsExport"

echo "Add NFS Client"
$CCSGEN --addresource nfsclient name="nfsClient" options="rw"
target="10.16.139.0/21"

echo "Add Service"
$CCSGEN --addservice ha-nfs-service domain="ccsFailoverDomain"
```

```
recovery="relocate"
# END ADDING RESOURCES

# BEGIN NESTING RESOURCES
echo "Add IP address as subservice of top level service"
$CCSGEN --addsubservice ha-nfs-service ip ref="$IPRESOURCE"

echo "Add filesystem as subservice of IP address"
$CCSGEN --addsubservice ha-nfs-service ip:fs ref="fileSystem"

echo "Add NFS export as subservice of filesystem"
$CCSGEN --addsubservice ha-nfs-service ip:fs:nfsexport ref="nfsExport"

echo "Add NFS client as subservice of NFS export"
$CCSGEN --addsubservice ha-nfs-service ip:fs:nfsexport:nfsclient
ref="nfsClient"
# END NESTING RESOURCES

# BEGIN CONFIGURE TWO NODE
echo "Configure for Two node"
$CCSGEN --setcman two_node=1 expected_votes=1
# END CONFIGURE TWO NODE

echo "Add xml tag to beginning of file"
sed -i '1 i <?xml version="1.0"?>' cluster.conf

# Call validate function
VALIDATE

echo

echo "The cluster.conf file has been generated"
echo

# Call DEPLOY function
DEPLOY

# End script logging
) | tee -a cluster_configuration/cluster_script_logs/cluster_output-$(date +
%F_%T).log
```

5 Cluster Administration

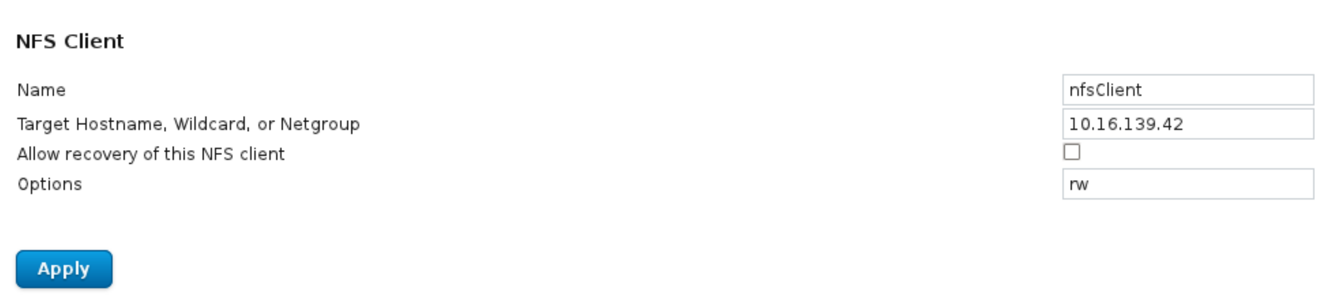
5.1 Adding Clients to NFS Export

This reference architecture provides an example of deploying NFS in a secure environment where access to the NFS share is explicit. There are two ways to allow access to NFS shares, the first way is to allow access to a network, or multiple hosts. The second way is to explicitly allow access to certain hosts on the network. By using the second method the NFS administrator has more control over which hosts can utilize the NFS services. This section demonstrates adding additional hosts to the clustered NFS service client using both *luci* and CCS.

Add hosts to NFS Service with Luci

Location: ha-luci

1. Get the current status of the NFS client export. Click on **Resources** on the upper navigation menu, then click on the NFS Client resource. As shown in **Illustration 5.1-1: Single Client Access** the NFS Client is providing access to a single host on the network.



NFS Client

Name	nfsClient
Target Hostname, Wildcard, or Netgroup	10.16.139.42
Allow recovery of this NFS client	□
Options	rw

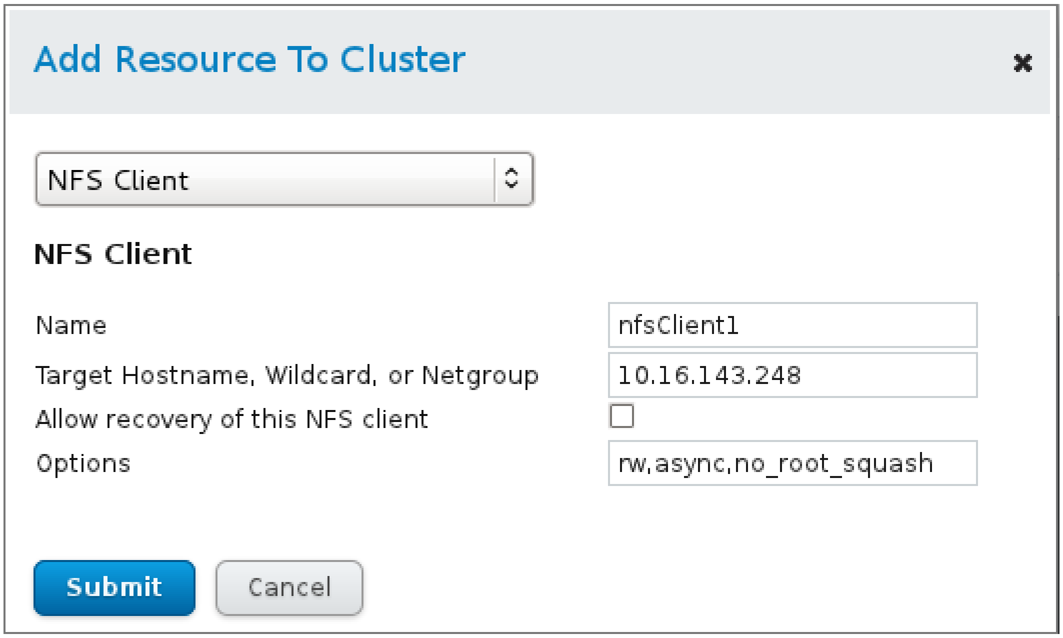
Illustration 5.1-2

Illustration 5.1-1: Single Client Access

2. The host *10.16.139.42* can mount the NFS share without problems. However, the host *10.16.143.248* can not access the NFS share.

```
# mount 10.16.139.46:/nfsshare /mnt/testdir/
mount.nfs: access denied by server while mounting 10.16.139.46:/nfsshare
```

3. Add the 10.16.143.248 host explicitly as a NFS client to the ha-nfs-service. Click on *Resources.* Then click ***Add*** and choose ***NFS Client*** from the dropdown menu. Add the following parameters as shown in **Illustration 5.1-3: Add NFS Client** and click ***Submit.***



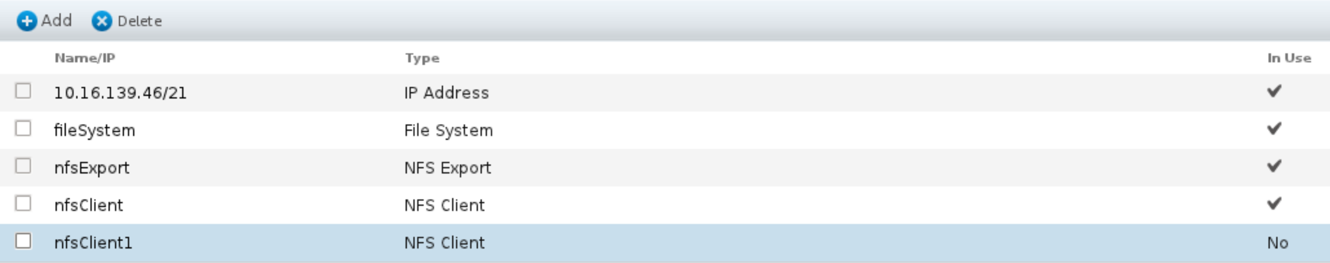
NFS Client

NFS Client

Name	nfsClient1
Target Hostname, Wildcard, or Netgroup	10.16.143.248
Allow recovery of this NFS client	□
Options	rw,async,no_root_squash

Illustration 5.1-3: Add NFS Client

4. The *nfsClient1* status is not in use as shown in **Illustration 5.1-4: NFS Client Not in Use**



	Name/IP	Type	In Use
□	10.16.139.46/21	IP Address	✓
□	fileSystem	File System	✓
□	nfsExport	NFS Export	✓
□	nfsClient	NFS Client	✓
□	nfsClient1	NFS Client	No

Illustration 5.1-4: NFS Client Not in Use

5. Associate *nfsclient1* to the ha-nfs-service. Click on *Service Groups*, click the *ha-nfs-service* and add the *nfsClient1* as a child resource to the *nfsExport* resource.

6. Check access from 10.16.143.248

```
# mount 10.16.139.46:/nfsshare /mnt/testdir/
```

7. The command returns without error

Add hosts to NFS Service with CCS

Location: 10.16.143.248

1. Try to mount the NFS share from host 10.16.143.248

```
# mount 10.16.139.46:/nfsshare /mnt/testdir/
mount.nfs: access denied by server while mounting (null)
```

Location: HA-LUCI

2. List the services on the cluster nodes

```
# ccs -h ha1 --lsservices
service: exclusive=1, domain=ccsFailoverDomain, name=ha-nfs-service,
recovery=relocate
  ip: ref=10.16.139.46/21
    fs: ref=fileSystem
      nfsexport: ref=nfsExport
        nfsclient: ref=nfsClient
resources:
  ip: sleeptime=10, address=10.16.139.46/21
  fs: device=/dev/mapper/EMCiSCSIp1, mountpoint=/nfsshare, name=fileSystem
  nfsexport: name=nfsExport
  nfsclient: target=10.16.139.42, name=nfsClient, options=rw
```

```
# ccs -h ha2 --lsservices
service: exclusive=1, domain=ccsFailoverDomain, name=ha-nfs-service,
recovery=relocate
  ip: ref=10.16.139.46/21
    fs: ref=fileSystem
      nfsexport: ref=nfsExport
        nfsclient: ref=nfsClient
resources:
  ip: sleeptime=10, address=10.16.139.46/21
  fs: device=/dev/mapper/EMCiSCSIp1, mountpoint=/nfsshare, name=fileSystem
  nfsexport: name=nfsExport
  nfsclient: target=10.16.139.42, name=nfsClient, options=rw
```

3. List the services that are confugred in the current local *cluster.conf* file

```
# ccs -f cluster.conf --lsservices
service: domain=ccsFailoverDomain, name=ha-nfs-service, recovery=relocate
  ip: ref=10.16.139.46/21
    fs: ref=fileSystem
      nfsexport: ref=nfsExport
        nfsclient: ref=nfsClient
resources:
  ip: sleeptime=10, address=10.16.139.46/21
  fs: device=/dev/mapper/EMCiSCSIp1, mountpoint=/nfsshare, name=fileSystem
  nfsexport: name=nfsExport
  nfsclient: target=10.16.139.42, name=nfsClient, options=rw
```

4. Here there is only one *nfsClient* as a child resource of the *nfsExport* resource

5. Add a new *nfsclient* called *nfsClient1* to the resources and associate it as a child service of *nfsExport*. Change into the directory that has the most current *cluster.conf* file. This modified file is pushed out to the cluster nodes.

```
# ccs -f cluster.conf --addresource nfsclient name="nfsClient1" \
options="rw,no_root_squash" target="10.16.143.248"
# ccs -f cluster.conf --addsubservice ha-nfs-service nfsclient
```

```
ref="nfsClient1"
```

6. List the updated services that are confugred in the current local *cluster.conf* file

```
# ccs -f cluster.conf --lsservices
service: domain=ccsFailoverDomain, name=ha-nfs-service, recovery=relocate
  ip: ref=10.16.139.46/21
    fs: ref=fileSystem
      nfsexport: ref=nfsExport
        nfsclient: ref=nfsClient
        nfsclient: ref=nfsClient1
resources:
  ip: sleeptime=10, address=10.16.139.46/21
  fs: device=/dev/mapper/EMCiSCSIp1, mountpoint=/nfsshare, name=fileSystem
  nfsexport: name=nfsExport
  nfsclient: target=10.16.139.42, name=nfsClient, options=rw
  nfsclient: target=10.16.143.248, name=nfsClient1, options=rw
```

2. Now the *nfsClient1* is nested at the correct level under the *nfsExport* resource.

3. Get the current version of the *cluster.conf* file off of HA1 and HA2 and the local *cluster.conf* file

```
# ccs -h ha1 --getversion
40
# ccs -h ha2 --getversion
40
# ccs -f cluster.conf --getversion
41
```

4. Synchronize the *cluster.conf* file with the cluster nodes

```
# ccs -f cluster.conf -h ha1.cloud.lab.eng.bos.redhat.com --setconf
# ccs -f cluster.conf -h ha2.cloud.lab.eng.bos.redhat.com --setconf
```

5. Get the current version of the *cluster.conf* file and list of services now supported from HA1 and HA2 file to ensure the *cluster.conf* file was propagated properly

```
# ccs -h ha1 --getversion
41
# ccs -h ha2 --getversion
41

# ccs -h ha1 --lsservices
service: domain=ccsFailoverDomain, name=ha-nfs-service, recovery=relocate
  ip: ref=10.16.139.46/21
    fs: ref=fileSystem
      nfsexport: ref=nfsExport
        nfsclient: ref=nfsClient
        nfsclient: ref=nfsClient1
resources:
 ip: sleeptime=10, address=10.16.139.46/21
 fs: device=/dev/mapper/EMCiSCSIp1, mountpoint=/nfsshare, name=fileSystem
 nfsexport: name=nfsExport
 nfsclient: target=10.16.139.42, name=nfsClient, options=rw
 nfsclient: target=10.16.143.248, name=nfsClient1, options=rw,no_root_squash

# ccs -h ha2 --lsservices
service: domain=ccsFailoverDomain, name=ha-nfs-service, recovery=relocate
```

```
  ip: ref=10.16.139.46/21
    fs: ref=fileSystem
      nfsexport: ref=nfsExport
        nfsclient: ref=nfsClient
        nfsclient: ref=nfsClient1
resources:
 ip: sleeptime=10, address=10.16.139.46/21
 fs: device=/dev/mapper/EMCiSCSIp1, mountpoint=/nfsshare, name=fileSystem
 nfsexport: name=nfsExport
 nfsclient: target=10.16.139.42, name=nfsClient, options=rw
 nfsclient: target=10.16.143.248, name=nfsClient1, options=rw,no_root_squash
```

10. Activate the new configuration on the nodes

```
# ccs -f cluster.conf -h ha1.cloud.lab.eng.bos.redhat.com --sync --activate
# ccs -f cluster.conf -h ha1.cloud.lab.eng.bos.redhat.com --sync --activate
```

11. Restart the ha-nfs-service

```
# clusvcadm -R ha-nfs-service
Local machine trying to restart service:ha-nfs-service...Success
```

Location: 10.16.143.248

1. Try to mount the NFS share from host 10.16.143.248

```
# mount 10.16.139.46:/nfsshare /mnt/testdir/

# mount | grep 46
10.16.139.46:/nfsshare on /mnt/testdir type nfs
(rw,vers=3,addr=10.16.139.46,clientaddr=10.16.143.248)
```

2. No errors were reported and the share is now mounted.

These activities can be scripted with a simple `for` loop to add multiple clients at one time. This is one of the great advantages of having a command line driven interface.

5.2 Managing a Cluster with CCS

With the addition of the CCS tool in Red Hat Enterprise Linux 6.1, a highly available cluster can be managed completely via command line interface. There are three main tools that will be used to accomplish this task as shown in **Table 5.2-1: Cluster Command Line Tools**

Tool	Role
ccs	Cluster Creation, Synchronization
clusvcadm	Cluster Service Administration
clustat	Cluster Status Utility

Table 5.2-1: Cluster Command Line Tools

CCS is used to update the *cluster.conf* file on all the cluster nodes and keep everything synchronized. CCS can be used to create a cluster, add services, *failover* domains, fence devices and resources. Once the cluster is configured it can be monitored with **clustat**. This tool is used to get the current status of the cluster, cluster nodes and the resources assigned to the cluster. Finally, **clusvcadm** is used to manage resources. For example, with **clusvcadm** resources can be stopped, started and migrated to and from nodes. These tools provide everything that is needed to fully manage the cluster without a graphical user interface.

This section will provide examples of managing and monitoring a cluster that was created with the CCS tool. Please refer to **4.3 Cluster Creation via CCS** for more information on how to create the cluster and cluster services with CCS.

Monitor the Cluster Status

Location: HA1

1. Start *clustat*

```
# clustat -i 3
Cluster Status for ccsCluster @ Thu Jun 16 13:03:58 2011
Member Status: Quorate

 Member Name                            ID   Status
 ------ ----                            ---- ------
 ha1.cloud.lab.eng.bos.redhat.com     1 Online, Local, rgmanager
 ha2.cloud.lab.eng.bos.redhat.com     2 Online, rgmanager

 Service Name             Owner (Last)                     State
 ------- ----             ----- ------                     -----
 service:ha-nfs-service   ha1.cloud.lab.eng.bos.redhat.com   started
```

2. From HA2, relocate the *ha-nfs-service*. to HA2

```
# clusvcadm -r ha-nfs-service -m ha2.cloud.lab.eng.bos.redhat.com
Trying to relocate service:ha-nfs-service to
ha2.cloud.lab.eng.bos.redhat.com...Success
service:ha-nfs-service is now running on ha2.cloud.lab.eng.bos.redhat.com
```

3. Monitor with **clustat** and ensure the service has been relocated

```
# clustat -i 3
Cluster Status for ccsCluster @ Thu Jun 16 13:09:28 2011
Member Status: Quorate

 Member Name                            ID   Status
 ------ ----                            ---- ------
 ha1.cloud.lab.eng.bos.redhat.com     1 Online, Local, rgmanager
 ha2.cloud.lab.eng.bos.redhat.com     2 Online, rgmanager

 Service Name             Owner (Last)                     State
 ------- ----             ----- ------                     -----
 service:ha-nfs-service   ha2.cloud.lab.eng.bos.redhat.com   started
```

Miscellaneous CCS Commands

1. Print the *cluster.conf* file on HA1

```
# ccs -h ha1 --getconf
<cluster config_version="36" name="ccsCluster">
  <fence_daemon/>
  <clusternodes>
    <clusternode name="ha1.cloud.lab.eng.bos.redhat.com" nodeid="1">
      <fence>
        <method name="HA1_Method_APC">
          <device name="Fence_APC_45" option="off" port="20" secure="on"/>
          <device name="Fence_APC_46" option="off" port="5" secure="on"/>

[ ... output abbreviated ... ]

    <service domain="ccsFailoverDomain" name="ha-nfs-service"
recovery="relocate">
      <ip ref="10.16.139.46/21">
        <fs ref="fileSystem">
          <nfsexport ref="nfsExport">
            <nfsclient ref="nfsClient"/>
          </nfsexport>
        </fs>
      </ip>
    </service>
  </rm>
</cluster>
```

2. Get the cluster version from HA1

```
# ccs -h ha1 --getversion
36
```

3. List the nodes in the cluster

```
# ccs -h ha1 --lsnodes
ha1.cloud.lab.eng.bos.redhat.com: nodeid=1
ha2.cloud.lab.eng.bos.redhat.com: nodeid=2
```

4. List the fence devices in the cluster

```
# ccs -h ha1 --lsfencedev
Fence_APC_45: power_wait=20, passwd=PASSWORD, ipaddr=10.16.128.45,
agent=fence_apc, login=refarch
Fence_APC_46: power_wait=20, passwd=PASSWORD, ipaddr=10.16.128.46,
agent=fence_apc, login=refarch
ha1_fence: passwd=PASSWORD, lanplus=on, ipaddr=10.16.41.230, auth=password,
agent=fence_ipmilan, login=root
ha2_fence: passwd=PASSWORD, lanplus=on, ipaddr=10.16.41.51, auth=password,
agent=fence_ipmilan, login=root
```

5. List the fence instances in the cluster

```
# ccs -h ha1 --lsfenceinst
ha1.cloud.lab.eng.bos.redhat.com
  HA1_Method_APC
    Fence_APC_45: port=20, secure=on, option=off
    Fence_APC_46: port=5, secure=on, option=off
    Fence_APC_45: port=20, secure=on, option=on
    Fence_APC_46: port=5, secure=on, option=on
  fence_ipmilan
```

```
    ha1_fence:
ha2.cloud.lab.eng.bos.redhat.com
  HA2_Method_APC
    Fence_APC_45: port=22, secure=on, option=off
    Fence_APC_46: port=3, secure=on, option=off
    Fence_APC_45: port=22, secure=on, option=on
    Fence_APC_46: port=3, secure=on, option=on
  fence_ipmilan
    ha2_fence:
```

6. List the *failover* domain in the cluster

```
# ccs -h ha1 --lsfailoverdomain
ccsFailoverDomain: restricted=0, ordered=0, nofailback=0
  ha1.cloud.lab.eng.bos.redhat.com:
  ha2.cloud.lab.eng.bos.redhat.com:
```

7. List the services in the cluster

```
# ccs -h ha1 --lsservices
service: domain=ccsFailoverDomain, name=ha-nfs-service, recovery=relocate
  ip: ref=10.16.139.46/21
    fs: ref=fileSystem
      nfsexport: ref=nfsExport
        nfsclient: ref=nfsClient
resources:
  ip: sleeptime=10, address=10.16.139.46/21
  fs: device=/dev/mapper/EMCiSCSIp1, mountpoint=/nfsshare, name=fileSystem
  nfsexport: name=nfsExport
  nfsclient: target=10.16.139.0/21, name=nfsClient, options=rw
```

5.3 Updating the Cluster

It is necessary to update the cluster nodes as new errata are released that provide performance enhancements, security updates and stability fixes. When updating a cluster environment it is necessary to perform certain steps to ensure there are no problems with the service availability.

Location: HA1

1. Confirm the *ha-nfs-service* is on HA1 and relocate the cluster services and stop the cluster software on HA1

```
# clustat
Cluster Status for ccsCluster @ Thu Jun 16 18:34:20 2011
Member Status: Quorate

 Member Name                       ID   Status
 ------ ----                       ---- ------
ha1.cloud.lab.eng.bos.redhat.com   1 Online, Local, rgmanager
ha2.cloud.lab.eng.bos.redhat.com   2 Online, rgmanager

Service Name                    Owner (Last)                       State
------- ----                    ----- ------                       -----
service:ha-nfs-service          ha1.cloud.lab.eng.bos.redhat.com   started

# service rgmanager stop
```

```
Stopping Cluster Service Manager:                          [  OK  ]

# ccs --stop -h ha1
```

2. Confirm that the *ha-nfs-service* is on HA2 and that the cluster services are stopped on HA1 and that the node is offline

```
# clustat
Cluster Status for ccsCluster @ Thu Jun 16 20:32:29 2011
Member Status: Quorate

 Member Name                         ID   Status
 ------ ----                         ---- ------
 ha1.cloud.lab.eng.bos.redhat.com    1 Offline
 ha2.cloud.lab.eng.bos.redhat.com    2 Online, Local, rgmanager

 Service Name             Owner (Last)                       State
 ------- ----             ----- ------                       -----
 service:ha-nfs-service   ha2.cloud.lab.eng.bos.redhat.com   started
```

3. Update HA1 and reboot the system since a new kernel is being installed.

```
# yum update
Loaded plugins: product-id, rhnplugin, subscription-manager
Updating Red Hat repositories.
Setting up Update Process
Resolving Dependencies
-->  Running transaction check
---> Package kernel.x86_64 0:2.6.32-131.4.1.el6 will be installed
---> Package kernel-firmware.noarch 0:2.6.32-131.2.1.el6 will be updated
---> Package kernel-firmware.noarch 0:2.6.32-131.4.1.el6 will be an update
---> Package openssl.x86_64 0:1.0.0-10.el6 will be updated
---> Package openssl.x86_64 0:1.0.0-10.el6_1.4 will be an update
-->  Finished Dependency Resolution

[ ... output abbreviated ... ]

Installing:
kernel           x86_64  2.6.32-131.4.1.el6   rhel-x86_64-server-6    23 M
Updating:
kernel-firmware noarch  2.6.32-131.4.1.el6  rhel-x86_64-server-6   2.5 M
openssl         x86_64  1.0.0-10.el6_1.4    rhel-x86_64-server-6   1.4 M

Transaction Summary
============================================================================
Install       1 Package(s)
Upgrade       2 Package(s)

Total download size: 27 M
Is this ok [y/N]: y
Downloading Packages:
----------------------------------------------------------------------------
Total
15 MB/s |  27 MB     00:01
Running rpm_check_debug
Running Transaction Test
```

```
Transaction Test Succeeded
Running Transaction
  Updating   : kernel-firmware-2.6.32-131.4.1.el6.noarch                1/5
  Installing : kernel-2.6.32-131.4.1.el6.x86_6                          2/5
  Updating   : openssl-1.0.0-10.el6_1.4.x86_64                          3/5
  Cleanup    : kernel-firmware-2.6.32-131.2.1.el6.noarch                4/5
  Cleanup    : openssl-1.0.0-10.el6.x86_64                              5/5
duration: 981(ms)
Installed products updated.

Installed:
kernel.x86_64 0:2.6.32-131.4.1.el6
Updated:
kernel-firmware.noarch 0:2.6.32-131.4.1.el6
openssl.x86_64 0:1.0.0-10.el6_1.4
Complete!

# reboot
```

4. When the node comes back up, make sure it is online

```
# clustat
Cluster Status for ccsCluster @ Thu Jun 16 21:00:36 2011
Member Status: Quorate

 Member Name                          ID   Status
 ------ ----                          ---- ------
 ha1.cloud.lab.eng.bos.redhat.com     1 Online, Local
 ha2.cloud.lab.eng.bos.redhat.com     2 Online

 Service Name              Owner (Last)                         State
------- ----               ----- ------                           -----
 service:ha-nfs-service     ha2.cloud.lab.eng.bos.redhat.com      started
```

5. Relocate *ha-nfs-service* to HA1 and make sure that the cluster services on HA2 are stopped.

```
# service rgmanager stop
Stopping Cluster Service Manager:                          [  OK  ]
# service cman stop
Stopping cluster:
   Leaving fence domain...                                 [  OK  ]
   Stopping gfs_controld...                                [  OK  ]
   Stopping dlm_controld...                                [  OK  ]
   Stopping fenced...                                      [  OK  ]
   Stopping cman...                                        [  OK  ]
   Waiting for corosync to shutdown:                       [  OK  ]
   Unloading kernel modules...                             [  OK  ]
   Unmounting configfs...                                  [  OK  ]

# clustat
Cluster Status for ccsCluster @ Thu Jun 16 21:07:00 2011
Member Status: Quorate

 Member Name                           ID   Status
 ------ ----                           ---- ------
 ha1.cloud.lab.eng.bos.redhat.com       1 Online, Local, rgmanager
 ha2.cloud.lab.eng.bos.redhat.com       2 Offline

 Service Name               Owner (Last)                     State
 ------- ----              ----- ------                      -----
 service:ha-nfs-service    ha1.cloud.lab.eng.bos.redhat.com  started
```

6. Update HA2

```
# yum update
```

7. Once the kernel update is installed, reboot the system and check that the node is back online and that the *ha-nfs-service* can relocate from one node to another.

```
# reboot
# clustat
# clusvcadm -r ha-nfs-service -m ha2.cloud.lab.eng.bos.redhat.com
Trying to relocate service:ha-nfs-service to
ha2.cloud.lab.eng.bos.redhat.com...Success
service:ha-nfs-service is now running on ha2.cloud.lab.eng.bos.redhat.com
# clustat
```

At this point both nodes have been updated with a new kernel and rebooted. The *ha-nfs-service* has been relocated from one node to another after the reboot and it is all working.

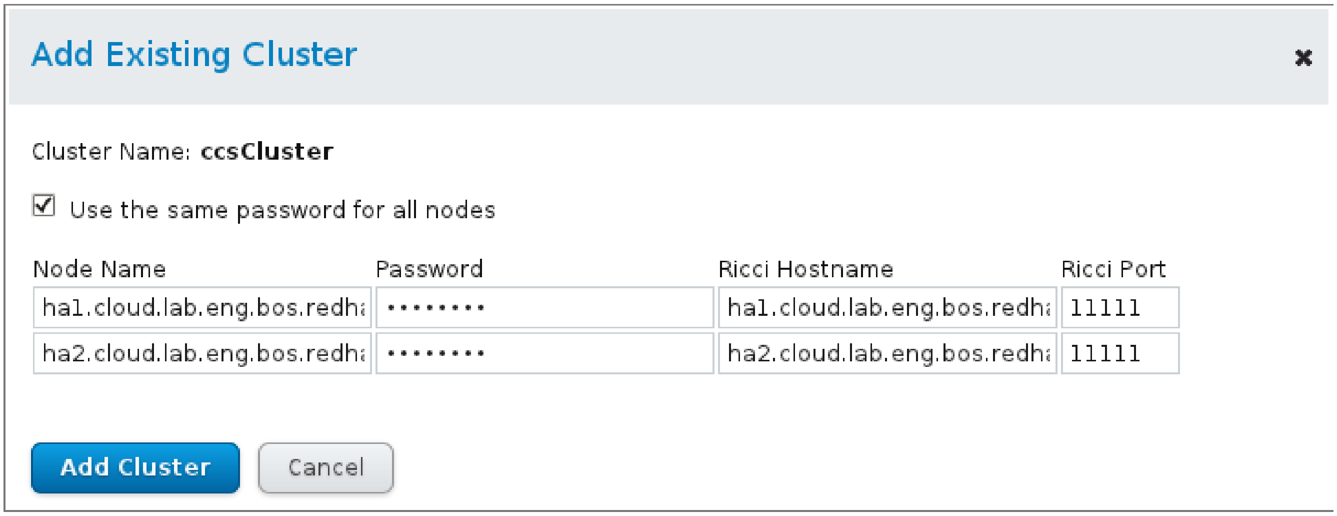
5.4 Importing a Cluster into Luci

Once the cluster is deployed with CCS, there are two methods to manage it. The first method is via **ccs**, **clustat** and **clusvcadm** and the second method is with *luci*. This section provides a high level overview of the process of importing a cluster into the *luci* management console. After just a few easy steps the cluster can be completely controlled using the graphical user interface. The only requirement is that there is a node on the network with a High Availability Management entitlement to RHN. At that point just issue a **`yum groupinstall "High Availability Management"`**, start the *luci* service and the node is ready to go.

Luci is a mature product that has been used in Red Hat High Availability products for quite some time now. Either method can be used to manage the cluster and both methods can be used simultaneously.

Location: Any System that has Browser Access to the Luci Server

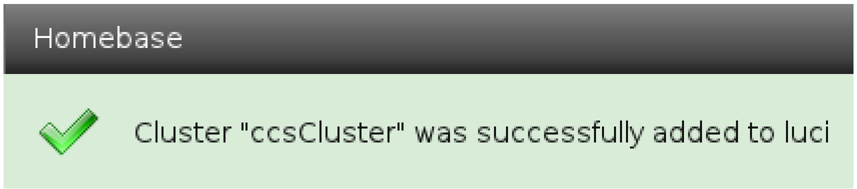
1. Open a browser and go to the luci interface and log in, click on ***Manage Clusters*** and click ***Add*** and populate the fields and click ***Add Cluster*** as shown in **Illustration 5.4-1: Importing Nodes into Luci** and **Illustration 5.4-2: Successful Cluster Import**



Cluster Name: **ccsCluster**

☑ Use the same password for all nodes

Node Name	Password	Ricci Hostname	Ricci Port
ha1.cloud.lab.eng.bos.redh	••••••••	ha1.cloud.lab.eng.bos.redh	11111
ha2.cloud.lab.eng.bos.redh	••••••••	ha2.cloud.lab.eng.bos.redh	11111

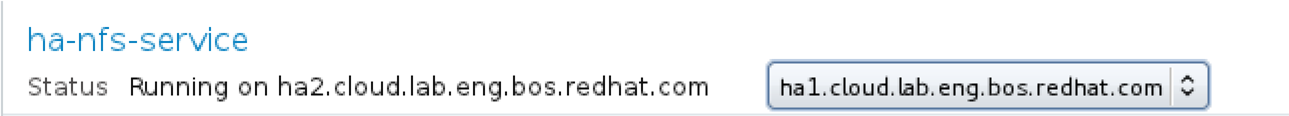


Homebase

Cluster "ccsCluster" was successfully added to luci

Illustration 5.4-2: Successful Cluster Import

2. Check that the resource can be migrated with the *Luci* interface. Click on **Service Groups** on the upper navigation menu, then click on the *ha-nfs-service* and migrate it to the other node as shown in **Illustration 5.4-3: Migrate Imported Resource**



ha-nfs-service

Status Running on ha2.cloud.lab.eng.bos.redhat.com

ha1.cloud.lab.eng.bos.redhat.com

6 Conclusion

This reference architecture has demonstrated how to build a two node cluster using standard hardware from the ground up. Both CCS and *luci* were used as ways to configure the cluster. This provides options based on the preference of the system administrator. The following deployment activities were performed:

- Cluster management node installation
- Cluster node installation
- Node security enforcement
- Cluster software deployment
- Cluster software configuration
- *Luci* and CCS administration tools

The utilization of the best practices in this reference architecture helps ensure that the NFS file system provides more up-time using this clustered environment.

Appendix A: Configuration Files

A.1 Multipath.conf file on HA1 and HA2

```
# This is a basic configuration file with some examples, for device mapper
# multipath.
# For a complete list of the default configuration values, see
# /usr/share/doc/device-mapper-multipath-0.4.9/multipath.conf.defaults
# For a list of configuration options with descriptions, see
# /usr/share/doc/device-mapper-multipath-0.4.9/multipath.conf.annotated

## By default, devices with vendor = "IBM" and product = "S/390.*" are
## blacklisted. To enable mulitpathing on these devies, uncomment the
## following lines.
#blacklist_exceptions {
#       device {
#               vendor  "IBM"
#               product "S/390.*"
#       }
#}

## Use user friendly names, instead of using WWIDs as names.
defaults {
        user_friendly_names no
}
##
## Here is an example of how to configure some standard options.
##
#
defaults {
#       udev_dir                /dev
#       polling_interval        10
#       selector                "round-robin 0"
#       path_grouping_policy    multibus
        getuid_callout          "/lib/udev/scsi_id --whitelisted
--device=/dev/%n"
#       prio                    alua
#       path_checker            readsector0
#       rr_min_io               100
#       max_fds                 8192
#       rr_weight               priorities
#       failback                immediate
#       no_path_retry           fail
#       user_friendly_names     yes
}
##
## The wwid line in the following blacklist section is shown as an example
## of how to blacklist devices by wwid.  The 2 devnode lines are the
## compiled in default blacklist. If you want to blacklist entire types
## of devices, such as all scsi devices, you should use a devnode line.
## However, if you want to blacklist specific devices, you should use
## a wwid line.  Since there is no guarantee that a specific device will
## not change names on reboot (from /dev/sda to /dev/sdb for example)
## devnode lines are not recommended for blacklisting specific devices.
##
blacklist {
#       wwid 26353900f02796769
        devnode "^(ram|raw|loop|fd|md|dm-|sr|scd|st)[0-9]*"
        devnode "^hd[a-z]"
}
multipaths {
        multipath {
                wwid                    36006048ccdaa201f61defbc5685c82b1
                alias                   EMCiSCSI
#               path_grouping_policy    multibus
#               path_checker            readsector0
#               path_selector           "round-robin 0"
#               failback                manual
#               rr_weight               priorities
#               no_path_retry           5
        }
#       multipath {
#               wwid                    1DEC_____321816758474
#               alias                   red
#       }
}
#devices {
#       device {
#               vendor                  "COMPAQ  "
#               product                 "HSV110 (C)COMPAQ"
#               path_grouping_policy    multibus
#               getuid_callout          "/lib/udev/scsi_id --whitelisted
--device=/dev/%n"
#               path_checker            readsector0
#               path_selector           "round-robin 0"
#               hardware_handler        "0"
#               failback                15
#               rr_weight               priorities
#               no_path_retry           queue
#       }
#       device {
#               vendor                  "COMPAQ  "
#               product                 "MSA1000         "
#               path_grouping_policy    multibus
#       }
#}

blacklist {
}
```

A.2 Kickstart file for cluster nodes

```
# Kickstart config file generated by RHN Satellite Config Management
# Profile Label : ha-cluster-nodes
# Date Created  : 2011-06-07 12:31:36.0

install
text
network --bootproto dhcp
url --url http://ra-ns1.cloud.lab.eng.bos.redhat.com/ks/dist/ks-rhel-x86_64-
server-6-6.1
lang en_US
keyboard us
zerombr
clearpart --all
bootloader --location mbr
timezone America/New_York
auth --enablemd5 --enableshadow
rootpw --iscrypted $1$bZaNCr5W$w0sdffHB1vM1bL.TzRm/
selinux --enforcing
reboot
firewall --disabled
skipx
key --skip
part /boot --fstype=ext3 --size=200
part pv.01 --size=1000 --grow
part swap --size=1000   --maxsize=2000
volgroup myvg pv.01
logvol / --vgname=myvg --name=rootvol --size=1000 --grow

%packages

@ Base

%pre

wget "http://ra-
ns1.cloud.lab.eng.bos.redhat.com/cblr/svc/op/trig/mode/pre/profile/ha-
cluster-nodes:1:RedHat" -O /dev/null


%pre
echo "Saving RHN keys..." > /dev/ttyS0
SYSTEM_ID=/etc/sysconfig/rhn/systemid
rhn_keys_found=no

insmod /lib/jbd.o
insmod /lib/ext3.o

if [ -f /lib/ext4.o ];
    insmod /lib/ext4.o
fi

mkdir -p /tmp/rhn
```

```
drives=$(list-harddrives | awk '{print $1}')
for disk in $drives; do
    DISKS="$DISKS $(fdisk -l /dev/$disk | grep -v "swap\|LVM\|Extended" |
awk '/^\/dev/{print $1}')"
done

# Try to find the keys on ordinary partitions
for disk in $DISKS; do
    name=test-$(basename $disk)
    mkdir -p /tmp/$name
    mount $disk /tmp/$name
    [ $? -eq 0 ] || continue # Skip to the next partition if the mount fails

    # Copy current RHN host keys out to be reused
    if [ -f /tmp/${name}$SYSTEM_ID ]; then
        cp -a /tmp/${name}$SYSTEM_ID /tmp/rhn
        rhn_keys_found="yes"
        umount /tmp/$name
        break
    fi
    umount /tmp/$name
    rm -r /tmp/$name
done

# Try LVM if that didn't work
if [ "$rhn_keys_found" = "no" ]; then
    lvm lvmdiskscan
    vgs=$(lvm vgs | tail -n +2 | awk '{ print $1 }')
    for vg in $vgs; do
        # Activate any VG we found
        lvm vgchange -ay $vg
    done

    lvs=$(lvm lvs | tail -n +2 | awk '{ print "/dev/" $2 "/" $1 }')
    for lv in $lvs; do
        tmpdir=$(mktemp -d findkeys.XXXXXX)
        mkdir -p /tmp/${tmpdir}
        mount $lv /tmp/${tmpdir} || continue # Skip to next volume if this
fails

        # Let's see if the keys are in there
        if [ -f /tmp/${tmpdir}$SYSTEM_ID ]; then
            cp -a /tmp/${tmpdir}$SYSTEM_ID /tmp/rhn/
            rhn_keys_found="yes"
            umount /tmp/${tmpdir}
            break # We're done!
        fi
        umount /tmp/${tmpdir}
        rm -r /tmp/${tmpdir}
    done

    # And clean up..
    for vg in $vgs; do
        lvm vgchange -an $vg
    done
fi

%post --nochroot
mkdir /mnt/sysimage/tmp/ks-tree-copy
if [ -d /oldtmp/ks-tree-shadow ]; then
cp -fa /oldtmp/ks-tree-shadow/* /mnt/sysimage/tmp/ks-tree-copy
elif [ -d /tmp/ks-tree-shadow ]; then
cp -fa /tmp/ks-tree-shadow/* /mnt/sysimage/tmp/ks-tree-copy
fi
cp /etc/resolv.conf /mnt/sysimage/etc/resolv.conf
cp -f /tmp/ks-pre.log* /mnt/sysimage/root/


%post --nochroot --interpreter /usr/bin/python
import xmlrpclib
import shutil
import os
import os.path
old_system_id = "/tmp/rhn/systemid"
new_system_id = "/mnt/sysimage/root/systemid.old"
try:

    new_keys = "1-fa21ca861a396a6326b93eed5cdd145e,1-ha-cluster-nodes"
    for key in new_keys.split(','):
        if key.startswith('re-'):
            os.exit(0)
    if os.path.exists(old_system_id):
        client =  xmlrpclib.Server("http://ra-
ns1.cloud.lab.eng.bos.redhat.com/rpc/api")
        key =
client.system.obtain_reactivation_key(open(old_system_id).read())
        f = open("/mnt/sysimage/tmp/key","w")
        f.write(key)
        f.close()
        shutil.copy(old_system_id, new_system_id)
except:
    # xml rpc due to  a old/bad system id
    # we don't care about those
    # we'll register those as new.
    pass


%post --logfile /root/ks-rhn-post.log
# --Begin RHN Satellite command section--
cat > /tmp/ssl-key-1 <<'EOF'
Certificate:

[ ... output abbreviated ... ]

-----END CERTIFICATE-----

EOF
# ssl-key1
cat /tmp/ssl-key-* > /usr/share/rhn/RHN-ORG-TRUSTED-SSL-CERT
```

```
perl -npe 's/RHNS-CA-CERT/RHN-ORG-TRUSTED-SSL-CERT/g' -i
/etc/sysconfig/rhn/*

mkdir -p /tmp/rhn_rpms/optional
cd /tmp/rhn_rpms/optional
wget -P /tmp/rhn_rpms/optional http://ra-
ns1.cloud.lab.eng.bos.redhat.com/download/package/cd0a8bf23c9cfe1d4f65fe492a
042cfd362497b2/0/1/12765/libxml2-python-2.7.6-1.el6.x86_64.rpm http://ra-
ns1.cloud.lab.eng.bos.redhat.com/download/package/845743006cf45c54957825906b
e4ca1ed726912d/0/1/19932/rhnlib-2.5.22-10.el6.noarch.rpm http://ra-
ns1.cloud.lab.eng.bos.redhat.com/download/package/004f79e54603dbab1a0408dc3d
e70c8687912d5b/0/1/12919/pyOpenSSL-0.10-2.el6.x86_64.rpm
rpm -Uvh --replacepkgs --replacefiles /tmp/rhn_rpms/optional/pyOpenSSL*
/tmp/rhn_rpms/optional/rhnlib* /tmp/rhn_rpms/optional/libxml2-python*
perl -npe 's|^(\s*serverURL\s*=\s*[^:]+://)[^/]*/|${1}ra-
ns1.cloud.lab.eng.bos.redhat.com/|' -i /etc/sysconfig/rhn/up2date

# now copy from the ks-tree we saved in the non-chroot checkout
cp -fav /tmp/ks-tree-copy/* /
rm -Rf /tmp/ks-tree-copy
# --End RHN Satellite command section--

# begin cobbler snippet
# begin Red Hat management server registration
mkdir -p /usr/share/rhn/
wget http://ra-ns1.cloud.lab.eng.bos.redhat.com/pub/RHN-ORG-TRUSTED-SSL-CERT
-O /usr/share/rhn/RHN-ORG-TRUSTED-SSL-CERT
perl -npe 's/RHNS-CA-CERT/RHN-ORG-TRUSTED-SSL-CERT/g' -i
/etc/sysconfig/rhn/*
key=""
if [ -f /tmp/key ]; then
    key=`cat /tmp/key`
fi

if [ $key ]; then
    rhnreg_ks --serverUrl=https://ra-ns1.cloud.lab.eng.bos.redhat.com/XMLRPC
--sslCACert=/usr/share/rhn/RHN-ORG-TRUSTED-SSL-CERT --activationkey=$key,1-
fa21ca861a396a6326b93eed5cdd145e,1-ha-cluster-nodes
else
     rhnreg_ks --serverUrl=https://ra-
ns1.cloud.lab.eng.bos.redhat.com/XMLRPC --sslCACert=/usr/share/rhn/RHN-ORG-
TRUSTED-SSL-CERT --activationkey=1-fa21ca861a396a6326b93eed5cdd145e,1-ha-
cluster-nodes
fi
# end Red Hat management server registration

# end cobbler snippet

rhn_check

# Start post_install_network_config generated code
# End post_install_network_config generated code
```

```
%post


# Start koan environment setup
echo "export COBBLER_SERVER=ra-ns1.cloud.lab.eng.bos.redhat.com" >
/etc/profile.d/cobbler.sh
echo "setenv COBBLER_SERVER ra-ns1.cloud.lab.eng.bos.redhat.com" >
/etc/profile.d/cobbler.csh
# End koan environment setup

# MOTD
echo >> /etc/motd
echo "RHN Satellite kickstart on $(date +'%Y-%m-%d')" >> /etc/motd
echo >> /etc/motd

# end of generated kickstart file

wget "http://ra-ns1.cloud.lab.eng.bos.redhat.com/cblr/svc/op/ks/profile/ha-
cluster-nodes:1:RedHat" -O /root/cobbler.ks
wget "http://ra-
ns1.cloud.lab.eng.bos.redhat.com/cblr/svc/op/trig/mode/post/profile/ha-
cluster-nodes:1:RedHat" -O /dev/null
```

A.3 Cluster.conf Configuration file

```
<?xml version="1.0"?>
<cluster config_version="138" name="ccsCluster">
        <clusternodes>
                <clusternode name="ha1.cloud.lab.eng.bos.redhat.com"
nodeid="1">
                        <fence>
                                <method name="HA1_Method_APC">
                                        <device name="Fence_APC_45"
option="off" port="20" secure="on"/>
                                        <device name="Fence_APC_46"
option="off" port="5" secure="on"/>
                                        <device name="Fence_APC_45"
option="on" port="20" secure="on"/>
                                        <device name="Fence_APC_46"
option="on" port="5" secure="on"/>
                                </method>
                                <method name="HA1_Method">
                                        <device name="HA1_Fence_Device"/>
                                </method>
                        </fence>
                </clusternode>
                <clusternode name="ha2.cloud.lab.eng.bos.redhat.com"
nodeid="2">
                        <fence>
                                <method name="HA2_Method_APC">
                                        <device name="Fence_APC_45"
option="off" port="22" secure="on"/>
                                        <device name="Fence_APC_46"
option="off" port="3" secure="on"/>
                                        <device name="Fence_APC_45"
option="on" port="22" secure="on"/>
                                        <device name="Fence_APC_46"
option="on" port="3" secure="on"/>
                                </method>
                                <method name="HA2_Method">
                                        <device name="HA2_Fence_Device"/>
                                </method>
                        </fence>
                </clusternode>
        </clusternodes>
        <cman expected_votes="1" two_node="1"/>
        <rm>
                <failoverdomains>
                        <failoverdomain name="ccsFailoverDomain"
nofailback="0" ordered="0" restricted="0">
                                <failoverdomainnode
name="ha1.cloud.lab.eng.bos.redhat.com"/>
                                <failoverdomainnode
name="ha2.cloud.lab.eng.bos.redhat.com"/>
                        </failoverdomain>
                </failoverdomains>
                <resources>
                        <ip address="10.16.139.46/21" sleeptime="10"/>
                        <fs device="/dev/mapper/EMCiSCSIp1"
mountpoint="/nfsshare" name="fileSystem"/>
                        <nfsexport name="nfsExport"/>
                        <nfsclient name="nfsClient" options="rw"
target="10.16.139.42"/>
                        <nfsclient name="nfsClient1"
options="rw,no_root_squash" target="10.16.143.248"/>
                </resources>
                <service domain="ccsFailoverDomain" name="ha-nfs-service"
recovery="relocate">
                        <ip ref="10.16.139.46/21">
                                <fs ref="fileSystem">
                                        <nfsexport ref="nfsExport">
                                                <nfsclient ref="nfsClient"/>
                                                <nfsclient
ref="nfsClient1"/>
                                        </nfsexport>
                                </fs>
                        </ip>
                </service>
        </rm>
        <fencedevices>
                <fencedevice agent="fence_apc" ipaddr="10.16.128.45"
login="refarch" name="Fence_APC_45" passwd="a22BCv" power_wait="20"/>
                <fencedevice agent="fence_apc" ipaddr="10.16.128.46"
login="refarch" name="Fence_APC_46" passwd="a22BCv" power_wait="20"/>
                <fencedevice agent="fence_ipmilan" auth="password"
ipaddr="10.16.41.230" lanplus="on" login="root" name="HA1_Fence_Device"
passwd="password"/>
                <fencedevice agent="fence_ipmilan" auth="password"
ipaddr="10.16.41.51" lanplus="on" login="root" name="HA2_Fence_Device"
passwd="password"/>
        </fencedevices>
</cluster>
```

A.4 NFS Configuration File

```
#
# Define which protocol versions mountd
# will advertise. The values are "no" or "yes"
# with yes being the default
#MOUNTD_NFS_V2="no"
#MOUNTD_NFS_V3="no"
#
#
# Path to remote quota server. See rquotad(8)
#RQUOTAD="/usr/sbin/rpc.rquotad"
# Port rquotad should listen on.
RQUOTAD_PORT=890
# Optinal options passed to rquotad
#RPCRQUOTADOPTS=""
#
#
# Optional arguments passed to in-kernel lockd
#LOCKDARG=
# TCP port rpc.lockd should listen on.
LOCKD_TCPPORT=891
# UDP port rpc.lockd should listen on.
LOCKD_UDPPORT=891
#
#
# Optional arguments passed to rpc.nfsd. See rpc.nfsd(8)
# Turn off v2 and v3 protocol support
#RPCNFSDARGS="-N 2 -N 3"
# Turn off v4 protocol support
#RPCNFSDARGS="-N 4"
# Number of nfs server processes to be started.
# The default is 8.
#RPCNFSDCOUNT=8
# Stop the nfsd module from being pre-loaded
#NFSD_MODULE="noload"
# Set V4 grace period in seconds
#NFSD_V4_GRACE=90
#
#
#
# Optional arguments passed to rpc.mountd. See rpc.mountd(8)
#RPCMOUNTDOPTS=""
# Port rpc.mountd should listen on.
MOUNTD_PORT=888
#
#
# Optional arguments passed to rpc.statd. See rpc.statd(8)
#STATDARG=""
# Port rpc.statd should listen on.
STATD_PORT=889
# Outgoing port statd should used. The default is port
# is random
#STATD_OUTGOING_PORT=2020
# Specify callout program
```

```
#STATD_HA_CALLOUT="/usr/local/bin/foo"
#
#
# Optional arguments passed to rpc.idmapd. See rpc.idmapd(8)
#RPCIDMAPDARGS=""
#
# Set to turn on Secure NFS mounts.
#SECURE_NFS="yes"
# Optional arguments passed to rpc.gssd. See rpc.gssd(8)
#RPCGSSDARGS=""
# Optional arguments passed to rpc.svcgssd. See rpc.svcgssd(8)
#RPCSVCGSSDARGS=""
#
# To enable RDMA support on the server by setting this to
# the port the server should listen on
#RDMA_PORT=20049
```

Appendix B: Appendix B: Troubleshooting Iptables

In this reference architecture, all the ports for the applications that are used are known. There are cases though, when implementing *iptables* rules where the port that is being accessed is unknown. One way to troubleshoot this is by using the logging option[13] provided by iptables. For this example, there is a NFS share being provided by NFSServer. NFSClient needs to issue a **showmount** command and find out what shares are available – the command fails.

```
# showmount --exports nfsserver
clnt_create: RPC: Port mapper failure - Unable to receive: errno 113 (No
route to host)
```

Location: NFSServer and NFSClient

1. List the existing rules on NFSServer

```
# iptables --numeric --verbose --list --line-number
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in   out  source
destination
1     2861  727K ha-nfs     all  --  *    * 0.0.0.0/0  0.0.0.0/0
2     2836  723K ha-cluster  all  -- *    * 0.0.0.0/0  0.0.0.0/0
3     1054  429K ACCEPT     all  --  *    * 0.0.0.0/0  0.0.0.0/0
state RELATED,ESTABLISHED
4       14  1176 ACCEPT     icmp --  *    * 0.0.0.0/0  0.0.0.0/0
5        0     0 ACCEPT     all  --  lo   * 0.0.0.0/0  0.0.0.0/0
6        0     0 ACCEPT     tcp  --  *    * 0.0.0.0/0  0.0.0.0/0
state NEW tcp dpt:22
7      220 83924 REJECT     all  --  *    * 0.0.0.0/0  0.0.0.0/0
reject-with icmp-host-prohibited

[ ... output abbreviated .. ]
```

2. Insert a logging option before the *REJECT* rule on the *INPUT* chain to place all output in */var/log/messages*

```
# iptables --input INPUT 7 --jump LOG --log-prefix "----IPTABLES-
REJECT------- "
# service iptables save
```

3. List the new rules on NFSServer

```
# iptables --numeric --verbose --list --line-number
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in   out   source
destination
1     3883  974K ha-nfs     all  --  *   *  0.0.0.0/0  0.0.0.0/0
2     3858  969K ha-cluster  all  -- *   *  0.0.0.0/0  0.0.0.0/0
3     1479  578K ACCEPT     all  --  *   *  0.0.0.0/0  0.0.0.0/0
state RELATED,ESTABLISHED
4       18  1512 ACCEPT     icmp --  *   *  0.0.0.0/0  0.0.0.0/0
5        0     0 ACCEPT     all  --  lo  *  0.0.0.0/0  0.0.0.0/0
```

```
6        0     0 ACCEPT     tcp  --  *    *  0.0.0.0/0  0.0.0.0/0
state NEW tcp dpt:22
7       42 17124 LOG        all  --  *    *  0.0.0.0/0  0.0.0.0/0
LOG flags 0 level 4 prefix `----IPTABLES-REJECT------- '
8      284  110K REJECT     all  --  *    *  0.0.0.0/0  0.0.0.0/0
reject-with icmp-host-prohibited
```

4. Issue the **showmount** command from NFSClient

```
# showmount --exports nfsserver
clnt_create: RPC: Port mapper failure - Unable to receive: errno 113 (No
route to host)
```

6. Test that the command works if *iptables* is stopped on HA1

    1. Stop *iptables* on *NFSServer*

```
# service iptables stop
iptables: Flushing firewall rules:                          [  OK  ]
iptables: Setting chains to policy ACCEPT: filter           [  OK  ]
iptables: Unloading modules:                                [  OK  ]
```

    2. Issue **showmount** from *NFSClient* again

```
# showmount --exports nfsserver
Export list for 10.16.139.43:
/nfsshare 10.16.139.42
```

5. Restart *iptables* on NFSServer and match the MAC address from NFSClient to the requests coming into NFSServer

```
# service iptables start
# tail --follow /var/log/messages | grep REJECT
Jun 10 21:29:26 ha2 kernel: ----IPTABLES-REJECT------- IN=bond0 OUT=
MAC=f0:4d:a2:3b:af:41:52:54:00:cc:1a:b8:08:00 SRC=10.16.139.42
DST=10.16.139.43 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=0 DF PROTO=UDP SPT=692
DPT=111 LEN=64
```

7. Match the IP and MAC addresses from NFSClient to the packets coming into NFSServer

```
# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UP qlen 1000
    link/ether 52:54:00:cc:1a:b8 brd ff:ff:ff:ff:ff:ff
    inet 10.16.139.42/21 brd 10.16.143.255 scope global eth0
    inet6 fe80::5054:ff:fecc:1ab8/64 scope link
       valid_lft forever preferred_lft forever
```

8. Step 5 shows that NFSClient requires UDP port 111 open for the **showmount** command to work.

9. Open UDP port 111 on NFSServer

```
# iptables --append ha-nfs --proto udp --dport 111 --jump ACCEPT
```

10. Issue another showmount command again and monitor the packets

```
# showmount --exports nfsserver
clnt_create: RPC: Port mapper failure - Unable to receive: errno 113 (No
route to host)
```

11. Go back to NFSServer and monitor the packets

```
# tail --follow /var/log/messages | grep REJECT
```

```
Jun 10 21:35:22 ha2 kernel: ----IPTABLES-REJECT------- IN=bond0 OUT=
MAC=f0:4d:a2:3b:af:41:52:54:00:cc:1a:b8:08:00 SRC=10.16.139.42
DST=10.16.139.43 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=6604 DF PROTO=TCP
SPT=699 DPT=33179 WINDOW=5840 RES=0x00 SYN URGP=0
```

12. Open UDP port 33179

```
# iptables --append ha-nfs --proto tcp --dport 33179 --jump ACCEPT
```

13. Go to NFSClient and issue the showmount

```
# showmount --exports nfsserver
Export list for 10.16.139.43:
/nfsshare 10.16.139.42
```

14. Now that the configuration is working, remove the logging rule and save the iptables rules on NFSServer.

```
# iptables --numeric --verbose --list --line-number
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source
destination
1     634K  335M ha-nfs     all  --  *      *       0.0.0.0/0
0.0.0.0/0
2     634K  335M ha-cluster  all  --  *      *       0.0.0.0/0
0.0.0.0/0
3     315K  293M ACCEPT     all  --  *      *       0.0.0.0/0
0.0.0.0/0           state RELATED,ESTABLISHED
4     4088  343K ACCEPT     icmp --  *      *       0.0.0.0/0
0.0.0.0/0
5        1    60 ACCEPT     all  --  lo     *       0.0.0.0/0
0.0.0.0/0
6        3   172 ACCEPT     tcp  --  *      *       0.0.0.0/0
0.0.0.0/0           state NEW tcp dpt:22
7       11  1230 LOG        all  --  *      *       0.0.0.0/0
0.0.0.0/0           LOG flags 0 level 4 prefix `----IP TABLES-REJECT-----'
8    10493 1088K REJECT     all  --  *      *       0.0.0.0/0
0.0.0.0/0           reject-with icmp-host-prohibited

# iptables --delete INPUT 7

# service iptables save
iptables: Saving firewall rules to /etc/sysconfig/iptables:[  OK  ]
```

Appendix C: Revisions

Revision 1.0	July 2011	Scott Collier
Initial Release		

[1] https://access.redhat.com

[2] https://access.redhat.com/kb/docs/DOC-30004

[3] http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html-single/Cluster_Administration/index.html#s1-iptables-CA

[4] http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Cluster_Administration/s1-iptables-CA.html

[5] http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html-single/Installation_Guide/index.html

[6] http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html-single/Installation_Guide/index.html

[7] https://access.redhat.com/kb/docs/DOC-48159

[8] https://access.redhat.com/kb/docs/DOC-53612

[9] https://access.redhat.com/kb/docs/DOC-53612

[10] https://access.redhat.com/kb/docs/DOC-8757

[11] https://access.redhat.com/kb/docs/DOC-53612

[12] https://access.redhat.com/kb/docs/DOC-30004

[13] http://magazine.redhat.com/2007/08/01/video-tip-from-rhces-firewalls/